X.509 Proxy Certificates for Dynamic Delegation

Von Welch,[1] Ian Foster,[2,3] Carl Kesselman,[4] Olle Mulmo,[5] Laura Pearlman,[4] Steven Tuecke,[2] Jarek Gawor,[2] Sam Meder,[3] Frank Siebenlist[2]

[1]National Center for Supercomputing Applications, University of Illinois
[2]Mathematics and Computer Science Division, Argonne National Laboratory
[3]Department of Computer Science, University of Chicago
[4]Information Sciences Institute, University of Southern California
[5]Royal Institute of Technology, Sweden

Contact author: Von Welch (vwelch@ncsa.uiuc.edu)

Abstract

Proxy credentials are commonly used in security systems when one entity wishes to grant to another entity some set of its privileges. We have defined and standardized X.509 Proxy Certificates for the purpose of providing restricted proxying and delegation within a PKI-based authentication system. We present here our motivations for this work coming from our efforts in Grid security, the Proxy Certificate itself, and our experiences in implementation and deployment.

1 Introduction

"Grids" [10] have emerged as a common approach to constructing dynamic, interdomain, distributed computing and data collaborations. In order to support these environments, Grids require a lightweight method for dynamic delegation between entities across organizational boundaries. Examples of these delegation requirements include granting privileges to unattended processes that must run without user intervention, the short-term sharing of files for collaboration, and the use of brokering services that acquire resources (e.g., storage, computing cycles, bandwidth) on behalf of the user.

The Globus Toolkit® [11] has emerged as the dominant middleware for Grid deployments worldwide. The Grid Security Infrastructure (GSI) [39,2,9] is the portion of the Globus Toolkit that provides the fundamental security services needed to support Grids. GSI provides libraries and tools for authentication and message protection that use standard X.509 public key certificates [5,16], public key infrastructure (PKI), the SSL/TLS protocol [6], and X.509 Proxy Certificates, an extension defined for GSI to meet the delegation requirements of Grid communities.

Proxy Certificates allow an entity holding a standard X.509 public key certificate to delegate some or all of its privileges to another entity, which may not hold X.509 credentials at the time of delegation. This delegation can be performed dynamically, without the assistance of a third party, and can be limited to arbitrary subsets of the delegating entity's privileges. Once acquired, a Proxy Certificate is used by its bearer to authenticate and establish secured connections with other parties in the same manner as a normal X.509 end-entity certificate.

Proxy Certificates were prototyped in early implementations of GSI. Subsequently, they have been refined through standardization in the IETF PKIX working group [17] and

have achieved RFC status. (At the time of this writing, the Proxy Certificate Internet draft [37] has passed IETF-wide public comment and is only awaiting assignment of an RFC number). GSI currently implements this standard.

GSI and Proxy Certificates have been used to build numerous middleware libraries and applications that have been widely deployed in large production and experimental Grids [2,3,4,19,35]. This experience has proven the viability of proxy delegation as a basis for authorization within Grids and has further proven the viability of using X.509 Proxy Certificates.

We start with a discussion of the requirements that spurred our use of X.509 public key certificates and motivated our development of Proxy Certificates. We follow with a technical description of the format of Proxy Certificates in Section 3. Section 4 describes how Proxy Certificates can be used to achieve single sign-on and delegation, and Section 5 describes how Proxy Certificates can be integrated with different types of authorization systems. Section 6 discusses current implementations and applications of Proxy Certificates. Section 7 discusses performance issues with Proxy Certificates and security tradeoffs in regard to those issues. We conclude with a discussion of related work in Section 8 and a summary in Section 9.

2 Motivation

We discuss first our motivation for the use of X.509 certificates and PKI as the basis for our GSI implementation. Then we discuss the motivations that led to the creation of Proxy Certificates as an enhancement to standard X.509 public key certificates.

2.1 Motivation for X.509 Certificates

GSI uses X.509 public key certificates and Secure Socket Layer (SSL) for authentication, not only because these are well-known technologies with readily available, well-tested open source implementations, but because the trust model of X.509 certificates allows an entity to trust another organization's certification authority (CA) without requiring that the rest of its organization do so or requiring reciprocation by the trusted CA.

This flexibility of trust model for X.509 certificates was a deciding factor between X.509 certificates and other common authentication mechanisms. For example, Kerberos [29] requires that all cross-domain trust be established at the domain level, meaning that organizations have to agree to allow cross-domain authentication, which can often be a heavyweight administrative process. In many common Grid deployments, only a few users and resources at a particular organization may participate in the Grid deployment, making the process of acquiring buy-in from the organization as a whole to establish the authentication fabric prohibitive.

2.2 Motivation for Proxy Certificates

The establishment of X.509 public key certificates and their issuing certification authorities provides a sufficient authentication infrastructure for persistent entities in Grids. However, several use cases exist that are not well covered by X.509 public key certificates alone.

- *Dynamic delegation*: Often, Grid users need to delegate some subset of their privileges to another entity on relatively short notice and only for a brief amount of time. For example, a user needing to move a dataset in order to use it in a computation may want to grant to a reliable file transfer service the necessary rights to access the dataset and storage so that it may perform a set of file transfers on the user's behalf. Since these actions may be difficult to predict, having to arrange delegation ahead of time through some administrator is prohibitive.
- *Dynamic entities*: In addition to delegation to persistent services and entities, the requirement exists to support delegation of privileges to services that are created dynamically, often by the user, that do not hold any form of identity credential. A common scenario is that a user submits a job to a computational resource and wants to delegate privileges to the job to allow it to access other resources on the user's behalf, for example, to access data belonging to the user on other resources or to start subjobs on other resources. An important point here is that the user wants to delegate privileges specifically to the job and not to the resource as a whole (i.e., other jobs being run by other users on the resource should not share the rights).
- *Repeated Authentication*: It is common practice to protect the private keys associated with X.509 public key certificates either by encrypting them with a pass phrase (if stored on disk) or by requiring a PIN for access (if on a smart card). This technique poses a burden on users who need to authenticate repeatedly in a short period of time, a situation that occurs frequently in Grid scenarios when a user is coordinating a number of resources.

A number of existing mechanisms could satisfy the first use case. For example, user-issued X.509 attribute certificates [8] could be used to delegate rights to other bearers of X.509 public key certificates. However, the heavyweight process of vetting associated with the issuing of public key certificates makes it prohibitive to use this method for the dynamically created entities described in the second use case: acquiring public key certificates for dynamically created, and often short-lived, entities would be too slow for practical use. Other means for authenticating these dynamic entities could have been used, for example bare keys as described in Section 8.5, but this approach would have required protocol modifications (or a new protocol) to accommodate the new authentication mechanism.

The third use case could be solved by caching the pass phrase or PIN required for access to the private key. However, this caching increases the risk of compromising the private key if the memory storing the pass phrase or PIN is somehow accessible to an attacker or is written out to disk (e.g., if it is swapped or in a core dump). In addition, reliably caching the PIN for a set of simultaneously running applications is a nontrivial software engineering task.

These requirements led us to develop an authentication solution that allows users to create identities for new entities dynamically in a lightweight manner, to delegate privileges to those entities (again in a dynamic, lightweight manner), to perform single sign-on, and to reuse existing protocols and software with minimal modifications. The result is the X.509 Proxy Certificate, which we describe in the following sections.

We note that while it may be possible to use Proxy Certificates for uses other than authentication, delegation, and message protection, for example the signing or encryption of long-lived documents, these other uses were not motivating factors in the Proxy Certificate design, and we have not investigated such use .

3 Description of Proxy Certificates

We now describe the contents of a Proxy Certificate and briefly discuss methods of revocation and path validation.

3.1 Proxy Certificate Contents

Proxy Certificates use the format prescribed for X.509 public key certificates [5,16] with the prescriptions described in this section on the contents. Proxy Certificates serve to bind a unique public key to a subject name, as a public key certificate does. The use of the same format as X.509 public key certificates allows Proxy Certificates to be used in protocols and libraries in many places as if they were normal X.509 public key certificates, which significantly eases implementation.

Unlike a public key certificate, however, the issuer (and signer) of a Proxy Certificate is identified by a public key certificate or another Proxy Certificate rather than a certification authority (CA) certificate. This approach allows Proxy Certificates to be created dynamically without requiring the normally heavyweight vetting process associated with obtaining public key certificates from a CA.

The subject name of a Proxy Certificate is scoped by the subject name of its issuer to achieve uniqueness. This is accomplished by appending a CommonName relative distinguished name component (RDN) to the issuer's subject name. The value of this added CommonName RDN should be at least statistically unique to the scope of the issuer. The value of the serial number in the Proxy Certificate should also be statistically unique to the issuer. Uniqueness for both of these values in our implementations is achieved by using the hash of the public key as the value. Unique subject names and serial numbers allow Proxy Certificates to be used in conjunction with attribute assertion approaches such as attribute certificates [8] and have their own rights independent of their issuer.

The public key in a Proxy Certificate is distinct from the public key of its issuer and may have different properties (e.g., its size may be different). As we describe in more detail in Section 4, except when using Proxy Certificates for single sign-on, the issuer does not generate the public key pair and has no access to the private key.

All Proxy Certificates must bear a newly defined critical X.509 extension, the Proxy Certificate Information (PCI) extension. In addition to identifying Proxy Certificates as such, the PCI extension serves to allow the issuer to express the desire to delegate rights to the Proxy Certificate bearer and to limit further Proxy Certificates that can be issued by that Proxy Certificate holder.

The issuer's desires about delegation to the Proxy Certificates bearer are expressed in the PCI extension by using a framework for carrying policy statements that allow for this delegation to be limited (perhaps completely disallowed). There exist a number of policy languages for expressing delegation policies (e.g., Keynote, XACML, XrML), instead of

defining a new mechanism or selecting a single existing policy language for expressing delegation policy (which probably would have bogged down the process of standardizing Proxy Certificates considerably), Proxy Certificates instead allow the issuer to use any delegation policy expression it chooses. The only restriction being that the issuer needs to know (through some out-of-band method) that the relying party understands its method of expression. This allows different deployments to select (or create) a method of delegation policy expression best suited for their purposes.

This use of arbitrary policy expressions is achieved through two fields in the PCI extension: a policy method identifier and a policy field. The policy method identifier is an object identifier (OID) that identifies the delegation policy method used in the policy field. The policy field then contains an expression of the delegation policy that has a format specific to the particular method (and may be empty for methods that do not require additional policy). For example, the identifier could contain an OID identifying the method as XACML; then the policy would contain an XACML policy statement.

The Proxy Certificate RFC defines two policy methods that must be understood by all implementations of Proxy Certificates (in addition to any more sophisticated methods they may implement):

- *Proxying*: This policy type indicates that the issuer of the Proxy Certificate intended to delegate all of their privileges to the Proxy Certificate bearer.
- *Independent*: This policy type indicates that the issuer of the Proxy Certificate intended the Proxy Certificate by itself to convey none of the issuer's privileges to the bearer. In this case the Proxy Certificate only serves to provide the bearer with a unique identifier, which may be used in conjunction with other approaches, such as attribute certificates, to grant its bearer privileges.

For both methods, the policy field is empty because the intended delegation policy is explicit in the type.

Table 1: Comparison of X.509 public key certificates and X.509 Proxy Certificates.

Certificate Attribute	X.509 Public Key Certificate	X.509 Proxy Certificates
Issuer/Signer	A certification authority	A public key certificate or another Proxy Certificate
Name	Any as allowed by issuer's policy	Scoped to namespace defined by issuer's name
Delegation from issuer	None	Allows for arbitrary policies expressing issuer's intent to delegate rights to Proxy Certificate bearer.
Key pairs	Uses unique key pair	Uses unique key pair

The PCI extension also contains a field expressing the maximum path lengths of Proxy Certificates that can be issued by the Proxy Certificate in question. A value of zero for this field prevents the Proxy Certificate from issuing another Proxy Certificate. If this field is not present, then the length of the path of Proxy Certificates, which can be issued by the Proxy Certificate, is unlimited.

3.2 Proxy Certificate Path Validation

Validation of a certificate chain has two distinct phases. First, validation of the certificate chain up to the public key certificate occurs, as described by RFC 3280 [16]. Validation of the Proxy Certificate portion of the chain is then performed as described in the Proxy Certificate RFC [37]. In summary these rules are as follows:

- Ensuring each Proxy Certificate has a valid Proxy Certificate Information extension as described in the previous section
- Ensuring each Proxy Certificate has a subject name derived from the subject name of its issuer
- Verifying the number of Proxy Certificates in the chain does not exceed the maximum length specified in any of the Proxy Certificate Information extensions in the chain
- Storing the delegation policies of each Proxy Certificate so that the relying party can determine the set of rights delegated to the bearer of the end Proxy Certificate used to authenticate

3.3 Revocation of Proxy Certificates

Currently there exists no implemented method for revocation of Proxy Certificates. The intent is that Proxy Certificates be created with short life spans, typically on the order of hours (with eight hours being the default of our implementation). Therefore, revocation has not been a pressing issue because this short lifetime limits the length of misuse if a Proxy Certificate is compromised. However, Proxy Certificates can be uniquely identified in the same manner as normal end-entity certificates, through the issuer and serial number, so the potential exists to revoke them by using the same mechanisms (e.g., CRLs [16] or OCSP [28]).

4 Use for Single Sign-on and Delegation

In this section we describe how Proxy Certificates can be used to perform single sign-on and delegation.

4.1 Enabling Single Sign-on

Normally the private key associated with a set of long-term X.509 credentials is protected in some manner that requires manual authentication on the behalf of its owner. While this process serves to provide a high level of protection of the private key, it can be prohibitively burdensome if the user needs to access the key frequently for authentication to other parties.

Proxy Certificates solve this problem by enabling single sign-on, that is, allowing the user to manually authenticate once, in order to create a Proxy Certificate that can be used repeatedly to authenticate for some period of time without compromising the protection on the user's long-term private key. This is accomplished by creating a new key pair (composed of a public and private key) and by subsequently using the user's long-term private key to create a short-lived Proxy Certificate. The Proxy Certificate binds the new public key to a new name and delegates some or all of the user's privileges to the new name. The Proxy Certificate and the new private key are then used by the bearer to authenticate to other parties. Since the Proxy Certificate has a short lifetime, it is typically permissible to protect the proxy certificate in a less secure manner than the long-term private key. In practice this means the Proxy Certificate private key is stored on a local file system and is protected only by local file system permissions; hence, the user's applications can access it without any manual intervention by the user.

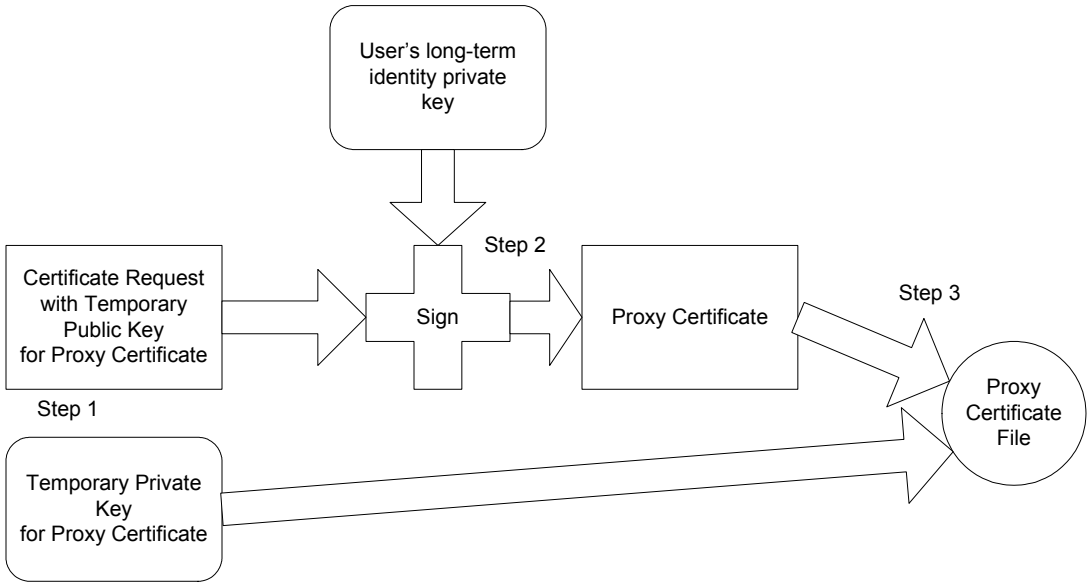

Figure 1: Creation of a Proxy Certificate for single sign-on. Steps are described in the text.

The process of creating a Proxy Certificate for single sign-on is shown in Figure 1. The steps, which are normally all done by a single application run by the user, are as follows:

1. A new key pair, consisting of a public and private key, is generated for use in the Proxy Certificate. The public key is encoded in a certificate request [20] for further processing.
2. The user's private key associated with their long-term public key certificate is accessed (possibly requiring the manual entering of a pass phrase or PIN by the user) to sign the certificate request containing the public key of the newly generated key pair, hence generating a Proxy Certificate. After the Proxy Certificate is signed, the user's long–term private key can remain secured (or the associated smart card can be removed) until the Proxy Certificate expires.
3. The Proxy Certificate and its associated private key are then placed in a file. This file is protected only by local file system permissions, to allow for easy access by the user.

When the Proxy Certificate expires, this process is repeated by the user to generate a new key pair and Proxy Certificate. The result from the perspective of the user is that manual authentication is required only infrequently to enable applications to authenticate on their behalf.

4.2 Delegation over a Network

Proxy Certificates can also be created so as to delegate privileges from an issuer to another party over a network connection without the exchange of private keys. This delegation process requires that the network connection be integrity-protected to prevent malicious parties from tampering with messages, but it does not require encryption, as no sensitive information is exchanged.

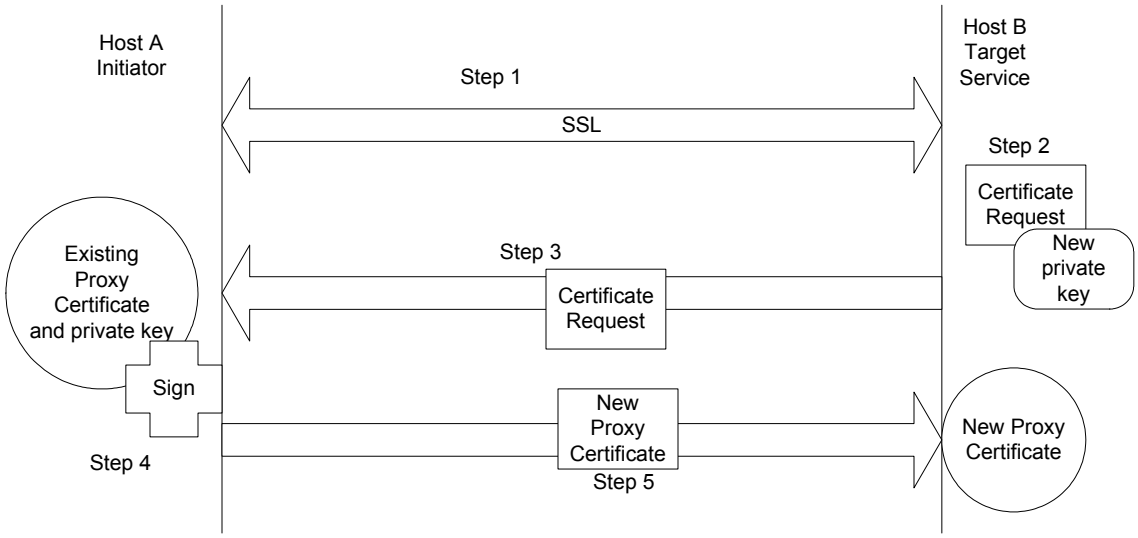

Figure 2: Delegation of a Proxy Certificate over a secured network connection. Steps are described in the text.

Figure 2 shows the steps involved in the delegation of privileges by creation of a Proxy Certificate over a network connection:

1. The initiator, on host A at left, connects to the target service on host B at right. The initiator and target service perform mutual authentication; the initiator uses its existing Proxy Certificate, and the target service uses the public key certificate of its own (not shown). After authentication, an integrity protected channel is established. These two steps can be accomplished by using the SSL protocol.
2. After the initiator expresses its desire to delegate by some application-specific means, the target service generates a new public and private key pair.
3. With the new public key, a signed certificate request is created and sent back over the secured channel to the initiator.
4. The initiator uses the private key associated with its own Proxy Certificate to sign the certificate request, generating a new Proxy Certificate containing the newly generated public key from the target service. The initiator fills in appropriate values for the fields in the Proxy Certificate, as described in Section 3.1, as well as a policy describing the rights it wishes to delegate.

5. The new Proxy Certificate is sent back over the secured channel to the target service, which places it into a file with the newly generated private key. This new Proxy Certificate is then available for use on the target service for applications running on the user's behalf.

While the host receiving the delegated Proxy Certificate may have a long-term key pair of its own (bound to an X.509 public key certificate that it used for authentication), this key pair is typically not reused for the delegated Proxy Certificate. The reason is that a given host may have multiple users delegating privileges to it that are intended to be bound to specific processes and not shared across processes. The generation of a new key pair for each process greatly simplifies the task of keeping privileges compartmentalized. This approach also allows a user to "revoke" the delegation by deleting the Proxy Certificate private key as described in [13]. In Section 7 we discuss the performance ramifications of this approach.

5 Authorization Models for Proxy Certificates

Proxy Certificates have three obvious modes of integration with authorization systems: full delegation of rights from the issuer—in effect, impersonation; no delegation of rights from the issuer, solely using attribute assertions to grant privileges; and a restricted delegation of some subset of the issuer's rights to the Proxy Certificate bearer. In this section we describe our experiences with each of these three methods.

5.1 Identity-based Authorization with Impersonation

In our initial implementations, we used Proxy Certificates almost exclusively as impersonation credentials that granted the bearer the full rights of the issuer. This approach has the advantage of integrating easily with identity-based authorization systems since these systems can simply treat the bearer of such a Proxy Certificate as they would its issuer. However, such usage is not ideal from the point of view of trying to achieve least privilege delegation because it supports only the full delegation of the issuer's rights. Thus, we explored other methods.

5.2 Proxy Certificates with Restricted Delegation

Proxy Certificates can be created with policies that delegate only a subset of the issuer's rights to the Proxy Certificate bearer. While this usage is more in line with the goal of enabling least privilege delegation, the implementation becomes more complex. As we described in Section 3.1, Proxy Certificates do not mandate any particular delegation language for the issuer to express a delegation policy, but instead provide a framework for containing policy statements using a method of the issuer's choosing.

The primary complication is that the relying party accepting the restricted Proxy Certificate must both understand the semantics of the delegation policy used and be able to enforce the restrictions that it imposes. Since these policies often contain application-specific restrictions, it is difficult for a security library handling the authentication of the Proxy Certificate to know what restrictions the application understands and is capable of enforcing. Without assurance that the application (or some other part of the software stack) will handle the enforcement of the restrictions, the authentication library cannot safely accept a restricted Proxy Certificate.

We have attempted to solve this problem by extending the API between the application and the security libraries to allow the application to express to the security libraries its knowledge and ability to handle given restriction delegation policies. This approach is difficult in practice, however, since it must be done on a per-application basis. For this reason, we have not used this form of Proxy Certificate authorization to a large degree.

5.3 Identity Creation with Additional Assertions

The third method of using Proxy Certificates in authorization systems is to have Proxy Certificates convey no rights to the bearer (i.e., a policy type of "independent" as described in Section 3.1) and then use attribute assertions to assign rights to the bearer. This method has the advantage that attributes may be granted to the bearer from a number of different sources and may be done so at times other than the creation of the Proxy Certificate.

However, two difficulties in implementation of this method have slowed our adoption:

- *Lack of protocol support:* The TLS protocol [6] and implementation of OpenSSL [32] (before the latest, version 0.9.7) lack support for X.509 attribute certificates. Thus, every application protocol must be modified to include a means of transporting attribute certificates. (Our recent move to a Web service-based protocol [39] may ease this burden.)
- *Lack of granularity in enforcement systems:* Many enforcement systems do not have the ability to enforce any policies with finer granularity than simple groups. Although there has been some work in finer-grained enforcement [25,33,12,21,22] these results are not yet portable across all applications and operating systems.

6 Proxy Certificate Implementations and Applications

Here we briefly describe our implementation of Proxy Certificates and some applications that use Proxy Certificates.

6.1 Implementation in Globus Toolkit's Grid Security Infrastructure

The Grid Security Infrastructure (GSI) implements Proxy Certificates to provide authentication and delegation capabilities for the Globus Toolkit. It allows application users to employ proxy certificates to authenticate to GSI-based services and to delegate Proxy Certificates to those services so that they may act on the user's behalf.

GSI is intended to work primarily with identity-based authorization systems and as such returns to the calling application an identity for the remote client. It is further intended to be used primarily with Proxy Certificates that have policies delegating the full set of their issuer's rights to their bearer. In this case it returns the subject name from the X.509 public key certificate that issued the original Proxy Certificate in the chain. As we describe in Section 6.4, GSI has also been used successfully with a combination of Proxy Certificates and attribute assertions. The use of GSI with restricted Proxy Certificates has been hampered by the issues described in Section 5.2.

GSI includes a GSS-API [24] library, which handles authentication and delegation using Proxy Certificates. This library is based heavily on the OpenSSL [32] library, an open

source implementation of the SSL protocol. The library uses OpenSSL to provide protocol support, including message protection and basic X.509 path validation. It adds to OpenSSL custom code for handling Proxy Certificates in addition to normal X.509 public key certificates and performing delegation.

6.2 MyProxy: An Proxy Certificate Repository

MyProxy [31,26] is a credential repository service that enables credential mobility and also alleviates the burden on users of managing and protecting files containing long-term secrets (i.e., private keys). We describe MyProxy briefly here, directing readers interested in more information to the references.

MyProxy is similar in function to a traditional credential repository as defined in the IETF SACRED working group [18]. By using Proxy Certificates, however, it can operate without long-term private keys ever leaving the MyProxy service. MyProxy allows a user to establish a protected channel to the MyProxy service using SSL (without a client-side certificate), to authenticate over that channel from a remote system using, for example, a username and pass phrase, and then obtain a Proxy Certificate bearing their privileges without having to carry their long-term public key certificate and private key around with them (a potentially error-prone and insecure process).

6.3 Use in Other Applications

The GSI libraries have also found uses in common applications. For example, Proxy Certificates can be used as an alternative authentication mechanism in secure shell (SSH) [15], CVS [14], and FTP [1]. These and other applications use the GSS-API library from GSI to allow a user to authenticate to an appropriate GSI-enabled daemon using their Proxy Certificate. The GSI-enabled SSH application also allows the user to delegate a new Proxy Certificate so that other GSI-enabled applications can be used on the remote system.

6.4 Proxy Certificates as Attribute Assertion Carriers

Combining public-key certificates with attribute assertions allow for the reuse of a single PKI across multiple application domains. In such a scenario, the PKI is used as a identity provider, and all applications or domain-specific privilege information (e.g., group memberships, clearance level, citizenship) is conveyed by separate attribute authorities.

As we mentioned in Section 5.3, however, many security protocols do not offer support for conveying attribute assertions. For example, the TLS protocol does not allow for attribute certificates in the set of provided client credentials. Thus, each application protocol must be modified to accommodate attribute assertions.

One way to circumvent this problem is by way of Proxy Certificates. When creating a Proxy Certificate, the proxy certificate issuer has the opportunity to add additional information to the proxy certificate by way of certificate extensions (in addition to the PCI extension described in Section 3.1). Several Grid projects use this technique to bundle application-specific attributes dynamically in the Proxy Certificate. The Community Authorization Service (CAS) [33,12] uses SAML authorization decisions [34] to assert that the identity may perform (a group of) actions on (a group of) objects. The VO Membership Service (VOMS) [38] is a role-based authorization system that uses

X.509 attribute certificates to assert a user's group membership(s), role(s), and capabilities. PRIMA [25] is a similar system that uses X.509 attribute certificates containing XACML [7] statements to assert a user's capabilities.

7 Performance and Security Issues

The expensive part of a Proxy Certificate creation is generating the new key pair. In this section, we consider only RSA key pairs, given the lack of support in commonly used open source software stacks for alternatives, such as elliptic curve cryptography (ECC) [27].

Generating an RSA public key pair involves finding a pair of suitable prime numbers, which is a nontrivial amount of work that furthermore scales exponentially with the key length. shows timings for key pair generation on a 2.8GHz Pentium 4 processor using the OpenSSL 0.9.7 library. We measure system CPU time and give averages over 100 keys.

Table 3: Key generation times for RSA key pairs

Size (bits)	Time (seconds)
512	0.040
768	0.094
1024	0.176
1536	0.415
2048	1.348

Unfortunately, use of specialized hardware such as cryptographic accelerators does not help these timings much, as such hardware is built with the assumption that RSA key generation occurs seldom and thus is not a performance-sensitive operation.

Consequently, key generation of normal key sizes may consume a substantial amount of CPU for hosts receiving delegated Proxy Certificates from multiple clients. It is tempting to use smaller key sizes because the lifetime of a Proxy Certificate key pair is comparably short. (Indeed, the 3.0 release of the Globus Toolkit does just this.) While a smaller key size may yet meet the targets for complexity necessary to make brute-force attacks infeasible within the short lifetime of the key pair, one must remember the cascading effects on the context in which such a key is used. For example, private data transferred during an FTP connection will typically remain sensitive long after the transfer is completed; and an eavesdropper who records the whole FTP transfer thus has a longer period of time than the life of the key pair during which to attack the protection it provided.

Thus, we note that Proxy Certificate generation comes with a nonnegligible penalty in server-side key generation. Currently this means that services must take appropriate precautions when accepting Proxy Certificate delegations, to prevent denial of service attacks. The development of solutions that mitigate this problem is left as future work.

8 Related Work

A number of schemes offer delegation in a similar manner to Proxy Certificates. We discuss a few of these schemes here and compare them to our Proxy Certificate work.

8.1 Kerberos V5

The Kerberos Network Authentication Protocol [23,29] 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" (TGTs) and support for delegation of rights via "forwardable" and "proxyable" tickets. The initial use of proxy credentials in Kerberos was described by Neuman [30], who also described restricted proxy credentials and proposed several uses for them, including cascaded delegation (using a proxy credential that contains restrictions to generate a new proxy with greater restrictions), authorization servers (servers that grant restricted proxy credentials based on a database of authorization information), and group servers (servers that grant restricted proxy credentials that convey rights to assert membership in groups).

From the perspective of a user, applications using Kerberos 5 are similar to applications using X.509 Proxy Certificates. The features of Kerberos 5 tickets formed the basis of many of the ideas surrounding X.509 Proxy Certificates. For example, the local creation of a short-lived Proxy Certificate 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 a Proxy Certificate can be delegated just as a forwardable ticket can be forwarded. Proxy Certificate and Kerberos also share the common method of protecting a TGT and protecting the private key of a Proxy Certificate by using local filesystem permissions.

The major difference between Kerberos TGTs and X.509 Proxy Certificates is that creation and delegation of a TGT requires the involvement of a third party (the Kerberos Domain Controller), while Proxy Certificates can be unilaterally created by their issuers without the active involvement of a third party.

8.2 X.509 Attribute Certificates

An X.509 attribute certificate (AC) [8] can be used to grant to a particular identity some attribute such as a role, clearance level, or alternative identity such as "charging identity" or "audit identity." Authorization decisions can then be made by combining information from the identity itself with signed attribute certificates providing binding of that identity to attributes. Attribute certificates can be issued either by a trusted entity specific to the issuance of attributes, known as an attribute authority, or by end entities delegating their own privileges.

In the case of an attribute authority, this method works equally well with attributes certificates bound to public key certificates or Proxy Certificates. For example, Proxy Certificates can be used to delegate the issuer's identity to various other parties who can claim attributes of the issuer. An AC could also be bound directly to a particular Proxy Certificate using the unique subject name from the Proxy Certificate.

The uses of ACs that are granted directly by end entities overlap considerably with the uses of Proxy Certificates. However, this AC-based solution to delegation has some disadvantages as compared to the Proxy Certificate-based solution:

- A similar modification to the validation framework, as in the Proxy Certificate RFC and described in Section 3.2, is needed in order to allow ACs to be signed by end entities.
- Identifying short-lived, dynamically created identities as described in Section 2.2, remains a unresolved problem.
- All protocols, authentication code, and identity-based authorization services must be modified to understand ACs.
- ACs must be created and signed by the long-term identity credentials of the end entity. This requirement implies that the entity must know in advance which other identities may be involved in a particular task in order to generate the appropriate ACs. On the other hand, Proxy Certificates bearers can delegate privileges through the creation of new Proxy Certificates without interaction of the entity holding the long-term identity credentials.

We believe there are many unexplored tradeoffs between ACs and Proxy Certificates. Reasonable arguments can be made in favor of either an AC-based solution to delegation or a Proxy Certificate-based solution to delegation. The approach to 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 SPX

SPX [36] uses a structure entitled a "ticket" for delegation and single sign-on that is similar in purpose to Proxy Certificates. The two mechanisms share many common features: the SPX ticket is combined with a private key to provide a set of credentials to provide the means for authentication; the ticket and its private key are short-lived and normally stored in a file protected by file permissions; and the implementation uses the GSS-API as the application interface.

The main difference is that SPX defines its own format for the ticket and its own protocols for authentication. Proxy Certificates, being based on X.509 public key certificates, allow for a significant reuse of the existing protocols and software designed for those certificates.

Proxy Certificates also include the concept of a delegation policy (Section 3.1), which allows for arbitrary delegation of subsets of the issuers rights to the Proxy Certificate bearer. In contrast, SPX tickets offer only an impersonation mode.

8.4 Delegation in Digital's DSSA

Gasser and McDermott [13] describe a delegation scheme used in Digital's Distributed System Security Architecture (DSSA). This restricted public-key based delegation is similar to Proxy Certificates in that it allows for cascading delegation, has delegations bound to unique keys, and has similar motivations. The primarily difference between Proxy Certificates and the DSSA work is our starting from X.509 public key certificates

in order to allow for maximum protocol and software reuse. It is also unclear to what extent the DSSA work has been implemented.

8.5 Future XML Alternatives

Proxy Certificates offer a pragmatic approach to delegation of rights in a SSL- and X.509-dominated world. By basing Proxy Certificates on the well-established X.509 certificates, the Proxy Certificates chains are easily exchanged in the SSL authentication protocol. Furthermore, by embedding the delegation policy statements inside of the Proxy Certificate, these delegation directives are exchanged as part of the SSL authentication process

At this time, we appear to be moving toward a Web services-dominated world. We envision that pure XML-based alternatives to SSL/TLS will be invented for authentication and key exchange based on new and emerging specifications and standards, such as XML-Signature, XML-Encryption, WS-Trust, and WS-SecureConversation. We expect these new standards to be more authentication mechanism agnostic and supporting alternatives to X.509, such as PGP, SPKI, or bare keys. Furthermore, these protocols are also expected to be able to communicate attribute and authorization assertions transparently without requiring modification of the application protocol. Some of our initial work in this area is described in [39].

We are investigating these XML-based technologies as alternatives or enhancements to Proxy Certificates. For example, the equivalent functionality of a Proxy Certificate could be achieved through a fine-grained SAML [34] authorization assertion expressed or an XACML policy statement that empowers a bare key. The generation of this key and the issuing of this authorization assertion could follow the same procedure and pattern as we use for Proxy Certificates.

9 Summary

Standard X.509 identity and attribute certificates allow for the static assignment of identities and rights. However, some environments require that end entities be able to delegate and create identities quickly. We have described Proxy Certificates, a standard mechanism for dynamic delegation and identity creation in public key infrastructures. Proxy certificates are based on X.509 public key certificates in order to allow for significant reuse of protocols and open source software. Our Grid Security Infrastructure (GSI) implementation of Proxy Certificates exploits these opportunities for reuse to provide a widely used implementation of Proxy Certificate mechanisms. A number of applications and widespread deployment demonstrate the viability of Proxy Certificate mechanisms.

Acknowledgments

We are pleased to acknowledge significant contributions to the Proxy Certificate RFC by David Chadwick, Doug Engert, Jim Schaad, and Mary Thompson. We are also grateful to numerous colleagues for discussions regarding Proxy Certificates, in particular Carlisle Adams, Joe Bester, Randy Butler, Keith Jackson, Steve Hanna, Russ Housley, Stephen Kent, Bill Johnston, Marty Humphrey, Sam Lang, Ellen McDermott, Clifford Neuman, and Gene Tsudik. Doug Engert coded the initial prototype implementation of Proxy

Certificates in GSI. Sam Meder, Jarek Gawor, and Sam Lang coded the current implementations. We also thank Jim Basney and the anonymous members of the program committee for reviewing and commenting on early versions of this paper.

This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under contracts W-31-109-Eng-38, DE-AC03-76SF0098, DE-FC03-99ER25397 and No. 53-4540-0080.

References

1. Allcock, B., et. al., Data Management and Transfer in High Performance Computational Grid Environments. *Parallel Computing Journal*, Vol. 28 (5), May 2002, pp. 749–771.
2. Butler, R., Engert, D. Foster, I. , Kesselman, C. , Tuecke, S. , Volmer, J. , and Welch, V. A National-Scale Authentication Infrastructure. *IEEE Computer*, 33(12):60–66, 2000.
3. Beiriger, J., Johnson, W., Bivens, H., Humphreys, S. and Rhea, R., Constructing the ASCI Grid. In *Proc. 9th IEEE Symposium on High Performance Distributed Computing*, 2000, IEEE Press.
4. Brunett, S., Czajkowski, K., Fitzgerald, S., Foster, I., Johnson, A., Kesselman, C., Leigh, J., and Tuecke, S., Application Experiences with the Globus Toolkit. In *Proc. 7th IEEE Symp. on High Performance Distributed Computing*, IEEE Press, 1998, 81–89.
5. CCITT Recommendation, X.509: The Directory – Authentication Framework. 1988.
6. Dierks, T., and Allen, C., The TLS Protocol Version 1.0, *RFC 2246*, IETF, 1999.
7. eXtensible Access Control Markup Language (XACML) 1.0 Specification, OASIS, February 2003.
8. Farrell, S., and Housley, R., An Internet Attribute Certificate Profile for Authorization, *RFC 3281*, IETF, April 2002.
9. Foster, I., Kesselman, C., Tsudik, G., and Tuecke, S., A Security Architecture for Computational Grids. In *Proc. ACM Conference on Computers and Security*, 1998, 83–91.
10. Foster, I. and Kesselman, C. Computational Grids. In I. Foster and C. Kesselman, eds., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 2–48.
11. Foster, I., and Kesselman, C. Globus: A Toolkit-Based Grid Architecture. In I. Foster and C. Kesselman, eds., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 259–278.
12. Foster, I., Kesselman, C. , Pearlman, L., Tuecke, S., and Welch, V., The Community Authorization Service: Status and Future. In *Proc. International Conference on Computing in High Energy Physics - CHEP 2003*.
13. Gasser, M., and McDermott, E., An Architecture for Practical Delegation in a Distributed System. In *Proc. 1990 IEEE Symposium on Research in Security and Privacy*, IEEE Press, 1990, 20–30.
14. gridCVS, *http://www.globus.org/gridcvs/*, 2002.
15. GSI-Enabled OpenSSH, *http://grid.ncsa.uiuc.edu/ssh/*, 2004.
16. Housley, R., Polk, W., Ford, W., and Solo, D., Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. *RFC 3280*, IETF, April 2002.
17. IETF Public-Key Infrastructure (X.509) (pkix) working group. *http://www.ietf.org/html.charters/pkix-charter.html*, January 2004.
18. IETF Securely Available Credentials (SACRED) working group. *http://www.ietf.org/html.charters/sacred-charter.html*, 2003.
19. Johnston, W. E., Gannon, D., and Nitzberg, B., Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid. In *Proc. 8th IEEE Symposium on High Performance Distributed Computing*, 1999, IEEE Press.

20. Kaliski, B., PKCS #10: Certification Request Syntax v1.5, *RFC 2314*, October 1997.
21. Keahey, K., Welch, V., Lang, S., Liu, B., and Meder. S., Fine-Grain Authorization Policies in the GRID: Design and Implementation. In *Proc. 1st International Workshop on Middleware for Grid Computing*, 2003, 170–177.
22. Keahey, K., and Welch, V. Fine-Grain Authorization for Resource Management in the Grid Environment. In *Proc. Grid2002 Workshop*, 2002, 199–206.
23. Kohl, J., and Neuman, C., The Kerberos Network Authentication Service (V5), *RFC 1510*, IETF, 1993.
24. Linn, J. Generic Security Service Application Program Interface, Version 2. *RFC 2078*, 1997.
25. Lorch, M., Adams, D., Kafura, D., Koneni, M., Rathi, A., and Shah, S., The PRIMA System for Privilege Management, Authorization and Enforcement in Grid Environments, *4th Int. Workshop on Grid Computing - Grid 2003*, November 17, 2003, Phoenix, AZ.
26. Lorch, M., Basney, J., and Kafura, D., A Hardware-secured Credential Repository for Grid PKIs. In *Proc. 4th IEEE/ACM International Symposium on Cluster Computing and the Grid*, Chicago, April 19–22, 2004 (to appear).
27. Menezs, A., *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.
28. Myers, M., Ankney, R., Malpani, A., Galperin, S., and Adams, C., X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, *RFC 2560*, IETF, June 1999.
29. Neuman, B. C., and Ts'o, T., Kerberos: An Authentication Service for Computer Networks. *IEEE Communications Magazine*, 32 (9), 33–88, 1994.
30. Neuman, B. C., Proxy-Based Authorization and Accounting for Distributed Systems. In *Proc. 13th International Conference on Distributed Computing Systems*, May 1993, 293–291.
31. Novotny, J., Tuecke, S., and Welch, V., An Online Credential Repository for the Grid: MyProxy. In *Proc. Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press, August 2001.
32. OpenSSL, *http://www.openssl.org*, 2002.
33. Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S., A Community Authorization Service for Group Collaboration. *IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002.
34. Security Assertion Markup Language (SAML) 1.1 Specification, OASIS, November 2003.
35. Stevens, R., Woodward, P., DeFanti, T., and Catlett, C. From the I-WAY to the National Technology Grid. *Communications of the ACM*, 40(11):50–61. 1997.
36. Tardo, J.J., and Alagappan, K., SPX: Global Authentication Using Public Key Certificates. In *Proc. 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, vol., Iss., 232–244.
37. Tuecke, S., Welch, V. Engert, D., Thompson, M., and Pearlman, L., Internet X.509 Public Key Infrastructure Proxy Certificate Profile, *draft-ietf-pkix-proxy-10 (work in progress)*, IETF, 2003.
38. VOMS Architecture v1.1, *http://grid-auth.infn.it/docs/VOMS-v1_1.pdf*, May 2002
39. Welch, V., Siebenlist, F., Foster, I., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Meder, S., Pearlman, L., and Tuecke, S., Security for Grid Services. In *Proc. Twelfth International Symposium on High Performance Distributed Computing (HPDC-12)*, IEEE Press 2003, 46.