Project Summary

As Grid computing is increasingly deployed and utilized in everyday scientific and (soon) business activities, the need increases for a persistent, user-centric, and flexible Grid security infrastructure. In particular, Grid entities (humans, organizations, and software artifacts) must be able to easily and safely access the *security credentials* under which they are authorized to operate. These Grid entities, or software acting on their behalf, must be able to dynamically manage and select the most appropriate (e.g., *least-privilege*) security credentials, whether they are operating as client or server. Many Grids today employ Public Key Infrastructure (PKI) for authentication. While PKI meets many of the Grid requirements it falls short of addressing many of the issues concerning private or secret key management, leaving the burden on the user to install and protect keys and certificates on the systems they may use.

This proposal builds on previous work we have done developing and deploying an online credential service, MyProxy. MyProxy was created in 2000 to solve the problem of credential delegation and was specifically targeted to support web portal requirements. MyProxy has been readily accepted by the Grid community and is deployed in Grids around the world, including the National Computational Science Alliance, National Partnership for Advanced Computational Infrastructure (NPACI), and NASA Information Power Grid (IPG), among others.

In this project, a three-phased approach builds on MyProxy version 0.4.2 to create the needed persistent credential management support for production middleware. In the first phase, MyProxy is partially redesigned to reflect lessons learned during our initial deployments, retested, hardened, more extensively documented, and packaged using the Grid Packaging Toolkit (GPT) for inclusion in the NMI Middleware release. This phase results in a true production-quality online credential repository. In the second phase, additional functionality is created, tested, and deployed to support multiple authentication mechanisms to MyProxy, support for per-user auditing of MyProxy actions, new support for per-user access control within MyProxy, and support for multiple credential types within MyProxy. In the third phase, longer-term research issues involving the Open Grid Services Architecture (OGSA) will be addressed and standards will be developed within the Global Grid Forum. Overall, this project uniquely expands Grid computing toward a national and international Grid infrastructure—both by making Grid computing *more secure* and by making Grid computing *more usable*.

1 Background and Motivation

The first year of the NSF Middleware Initiative (NMI) produced the critical first steps toward establishing a national- and international-scale persistent infrastructure that builds upon the ubiquity of lower-level protocols such as the Internet Protocol (IP) by adding higher-level support for security, information services, and resource management. The security aspects of the NMI effort were largely focused on the Grid Security Infrastructure (GSI) [16], which is a core component of the Globus toolkit [17]. GSI supports a Public Key Infrastructure (PKI), building on the standard GSS API [28], SSL/TLS [13] protocol, and X.509 [25] certificates to provide secure single sign-on, credential delegation, and simple resource authorization control. GSI is an excellent foundation upon which to build Grids that are based on the use of Certificate Authorities (CAs), whereby a trusted agent attests to the binding between a public key and some Grid entity (primarily humans and machines). GSI particularly excels in a style of computation that is familiar to most first-time Grid users, a user "logs on" to the Grid in the morning, performs some Grid computations through the course of the day, and then leaves the Grid at the close of the day. The defining aspects of this style of Grid interaction are:

- User Grid computations do not generally persist beyond the time that the users leave at night.
- A user's single security credential is sufficient to authenticate to all resources in the Grid.
- All of a user's computations originate through the login shell from which the user performed a Grid sign-on operation.
- Generally, a user's computations act as clients rather than (long-lived) servers.

However powerful this style of Grid computing may be, there are challenges that must be addressed to make Grid computing truly revolutionary and ubiquitous. Users must be able to originate computation easily and securely from a number of locations and devices. Users must be able to use multiple authentication techniques (e.g., GSI, Kerberos [34], Passport [29], etc.) to engage Grid services—some users will want to carefully hand-select the authentication protocol and credential, while some users will want this to be completely transparent. Users must be able to easily set up their own (temporary) services for consumption by the Grid community and have these services persist for weeks or longer. In short, a robust, secure, and flexible on-line credential repository must be made available through the NMI program.

The focus of this project is the continued development of the MyProxy Online Credential Repository [36] for inclusion in the NMI Middleware release. MyProxy has been an active development project at the National Center for Supercomputing Applications (NCSA) for approximately two years, and MyProxy software has been deployed in the National Computational Science Alliance (Alliance), National Partnership for Advanced Computational Infrastructure (NPACI), and NASA Information Power Grid (IPG).

MyProxy will be partially redesigned to reflect lessons learned during our initial deployments, retested, hardened, more extensively documented, and packaged using the Grid Packaging Toolkit (GPT) for inclusion in the NMI Middleware release. This will primarily be performed in the first 3 months of the project. The remainder of the two-year effort will be used to develop additional functionality in the MyProxy core, followed by new releases into the NMI software repository. The protocols of an online credential repository will also be formalized and standardized in the Global Grid Forum (GGF) [20] through the efforts of PI Basney, who is a key

contributor in the existing GSI Working Group, and co-PI Humphrey, who is a member of the steering committee of the GGF and a co-director (with Steve Tuecke) of the Security Area of the GGF. Through NMI support, MyProxy will provide a necessary and robust functionality for emerging national and international Grids, including Grids that encompass Web Services through the support of the newly-announced Open Grid Services Architecture (OGSA) [18].

The rest of this proposal is organized as follows. Section 1.1 contains a more detailed discussion of the requirements of an online credentials repository and gives an overview of the existing MyProxy mechanism. Section 1.2 describes the related work. Section 2 presents the proposed work, starting with the integration of MyProxy with NMI middleware distribution(Section 2.1) and then describing the additional needed near-term functionality for MyProxy in Section 2.2: the support for additional authentication mechanisms (Section 2.2.1); the support for auditing, both for the system (Grid) administrator and the individual users (Section 2.2.2); improved access control (Section 2.2.3); and improved support for multiple credentials within MyProxy (Section 2.2.4). Section 2.3 contains longer-term goals, specifically the description of our standardization efforts and how MyProxy will support OGSA-compliant Grids. Section 3 contains the schedule and milestones. Section 4 contains the broader impact of the proposed work, and Section 5 contains the summary.

1.1 MyProxy Online Credential Repository

An online credential repository (OCR) is a key component in a middleware security infrastructure. Acquiring valid credentials is a prerequisite to accessing secure services. In a Public Key Infrastructure (PKI) such as GSI, users obtain long-term credentials from a Certificate Authority (CA). This general procedure is shown in Figure 1. To start this process, a user typically runs a special tool from her workstation to generate a public-private keypair. Only the public key is presented to the CA; in many situations the public key is emailed to the CA,

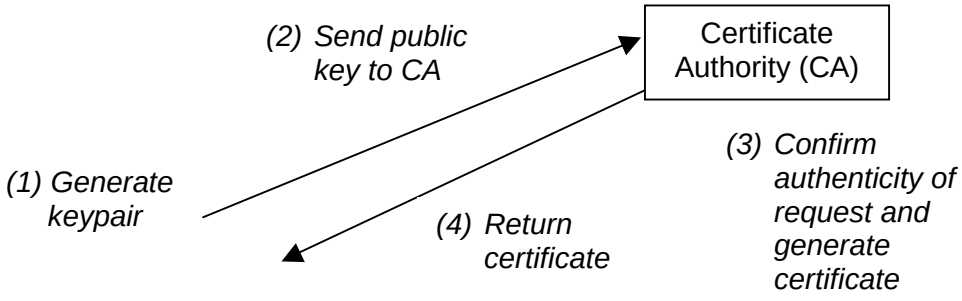

Figure 1: Obtaining a Certificate from a Certificate Authority

but, under more stringent security conditions, the public key can be required to be hand presented to the CA, with a photo ID the confirms the user's identity. Upon receiving this request, the CA performs checks that are a function of the level of assurance being granted by the certificate. These checks can often be a simple challenge-response to confirm the validity of the email address presented to the CA, but could be as stringent as requiring that a national passport be presented to the CA. After passing these tests, the CA generates and digitally signs a certificate (with the private key of the CA) that, in essence, binds the public key to a unique identifier representing the entity that generated the keypair. As long as the CA is trusted, a "relying party" (an entity that must base some action on the validity of the CA's certificates) believes that a particular entity is associated with a particular public key. Of course, the relying

party must engage in some security protocol (typically the SSL/TLS Handshake protocol [13]) to confirm that the party that asserts that it *is* a particular entity has access to the private key that corresponds to the public key in the certificate. Relying parties either have direct access to the CA's public key (perhaps the relying parties are running software in which the public key—and thus trust in a CA—is directly embedded via some configuration files) or a CA's public key can itself be asserted by a different CA (in which case a relying party must ultimately be configured to trust a CA in this chain).

Deploying a Certificate Authority to certify a community of users is a significant challenge in itself, and there have been considerable efforts in the Global Grid Forum to determine *best practices* for running a CA [8]. However, the credential management challenge does not end once a user has his or her keys and signed certificate. Users must have convenient access to their credentials on different computers, and the credentials must be kept safe from loss and secure from disclosure. The default mechanism by which to store credentials is in the file system local to the machine of a user. Storing credentials in files in user accounts on a general-purpose file system has three significant drawbacks. First, the file system may not provide adequate protection against key loss or compromise. Second, accessing the file system from a remote computer can be inconvenient and insecure. Third, users may have multiple credentials to enable access to secure services with different security requirements, and managing the different credential files directly is cumbersome and error-prone.

An OCR provides a credential storage mechanism that can be both more secure and more convenient than a general-purpose file system. MyProxy is a particular instantiation of an OCR and currently allows users to upload GSI credentials for later retrieval. Authentication in GSI is based on proxy credentials [41]. A proxy credential consists of a proxy certificate and an associated private key. The proxy certificate is an X.509 certificate that is derived from a standard X.509 end entity certificate or another proxy certificate and signed by the private key associated with the source certificate. Proxy credentials serve to limit the vulnerability of the end entity private key while supporting GSI requirements for single sign-on and credential delegation. Rather than entering a pass phrase to decrypt the private key on every authentication operation, or stashing the pass phrase or unencrypted key on the local system for repeated use, the user can use the private key once to create a proxy credential. The proxy certificate contains restrictions, such as a short lifetime, that limit the vulnerability if the proxy key should be compromised. The proxy key can then be stashed unencrypted for the duration of the user's session.

Proxy credentials can also be used to delegate credentials to processes acting on the end entity's behalf without transferring the end entity's private key to the process. Instead, the process generates its own proxy certificate and key and asks the delegating entity to sign the certificate, thereby allowing credentials to be forwarded over the network without transferring private keys. The delegating entity will typically place restrictions in the proxy certificate to limit the vulnerability of the delegated credential. Determining what to restrict is a research topic [40].

MyProxy uses delegation to upload and download credentials, so private keys are never transferred over the network. To upload a credential to MyProxy, the client delegates a proxy credential to the MyProxy server. To retrieve a credential, the client asks the MyProxy server to delegate a proxy credential to it.

In the current version of MyProxy, users associate a username and pass phrase with the credential during the upload operation. Users can then use a MyProxy client program to retrieve the credential at a later time, from a different machine, by authenticating with the username and pass phrase, allowing users to obtain credentials when they need them from any machine, without manually transferring credential files. Figure 2 presents an overview of

different uses of MyProxy. The left side of the figure shows a user uploading a proxy credential from one computer to the MyProxy server and then fetching a proxy credential from the MyProxy server on another computer. The user can then use the credential to submit jobs to schedulers and access Grid resources.

Note that the security of the credential itself is not weakened in this system, because the pass phrase is not sent in cleartext, and the original credential itself was only protected via pass phrase on the local file system. Users typically retrieve a proxy credential with a short lifetime, valid only for the duration of the current session, so rather than storing copies of long-term credentials on the different machines used, the user just retrieves a short-term credential on demand. This significantly reduces the vulnerability of the user's credentials because credentials are distributed on an as-needed basis and quickly expire after they are needed, so the user does not have long-lived credentials distributed across the Grid. A single MyProxy server can support a single user, a project within an organization, an entire organization, or a virtual community.

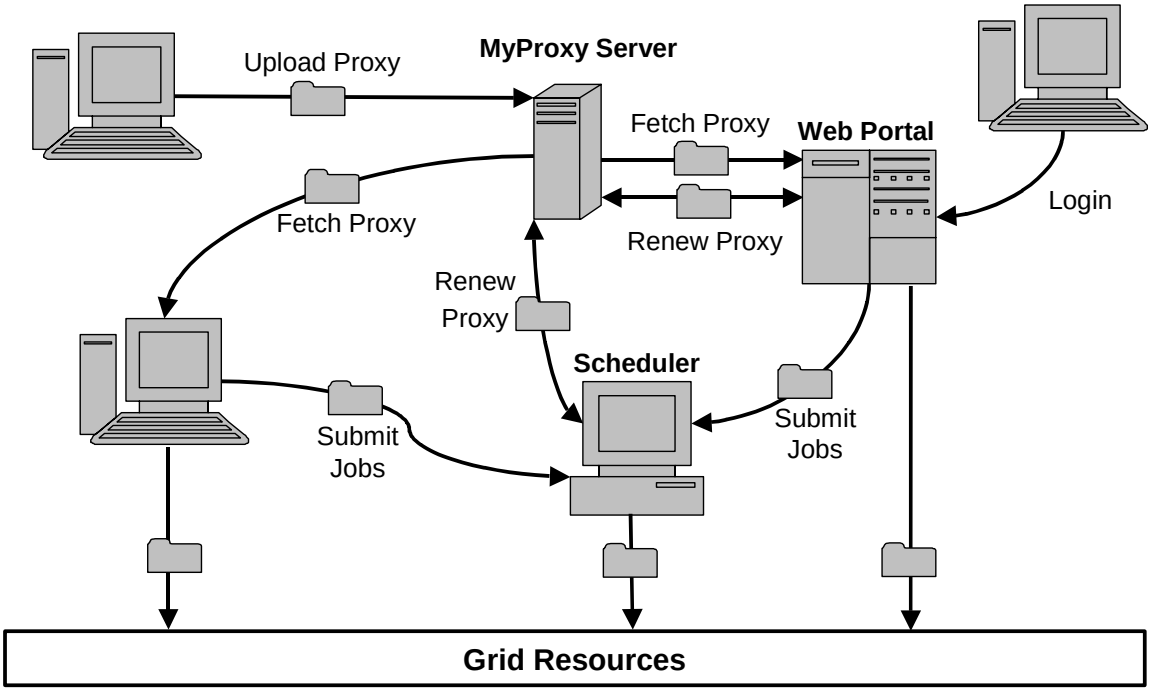

Figure 2: Using MyProxy to Access Grid Resources

The initial motivation for MyProxy was to support proxy delegation to web servers that act as portals to Grid services [36]. These systems require credentials to access Grid services on the user's behalf. However, standard web technologies do not provide adequate support for delegating credentials directly to web servers. Also, the user's credentials are not guaranteed to be directly available on the web browser machine. Therefore, to delegate credentials to a portal, users first delegate their credentials to the MyProxy server. Then, as shown on the right side of Figure 2, when a user accesses a portal, he or she enters the name of the MyProxy server and the username and pass phrase under which the credentials were stored. The portal can then use the username and pass phrase to retrieve a credential for the user from the MyProxy server. The portal and MyProxy server perform TLS mutual authentication using X.509 service credentials, so the portal knows it is communicating with the correct MyProxy server and the MyProxy server can verify that the portal is authorized to retrieve the user's credentials. The portal retrieves the user’s proxy, which is then used as the basis for

subsequent authentication to schedulers and Grid resources.

MyProxy has been recently extended to support credential renewal. When a user delegates a credential to a service, so it can act on the user's behalf, she may be unable to predict how long a valid credential will be needed. For example, when submitting a batch job to a queuing service such as PBS [38], the user cannot predict how long the job will wait in the queue and may not be able to accurately predict how long it will run. Rather than delegating a credential with the maximum expected lifetime needed, the user can delegate a renewable credential with a shorter lifetime. Thus, rather than delegating long-lived credentials to many services, the user can limit the vulnerability of his or her credentials by storing a long-lived credential in the well-secured MyProxy repository and only delegating short-lived credentials to other, less-trusted services. When a service needs to renew a credential before it expires, it authenticates to MyProxy and requests a new credential. Figure 2 shows both the scheduler and web portal using MyProxy's credential renewal feature. Discussions are underway with the Condor Team at the University of Wisconsin to add credential renewal functionality to Condor-G [19].

In summary, MyProxy is currently being used in production installations, primarily as a mechanism for Web-based, GSI-based, Grid activities. Its current version is an excellent candidate for (re-packaged) inclusion into the NMI Middleware distribution. Additional functionality (described in Section 2.2) is needed to support more flexible usage scenarios facilitated by multi-organizational and multi-national Grids built on an increasingly wide range of underlying (security) technologies.

1.2 Related Work

MyProxy provides, at the most basic level, a facility by which authorized clients can securely retrieve credentials on demand over the network. Two types of online credential retrieval services are possible: (1) an online credential repository (OCR), such as MyProxy, which stores credentials created by a trusted authority, and (2) an online credential authority, which creates new credentials on demand. Credential repositories can be further categorized by whether they handle the credentials as opaque objects or if they are able to interpret or interact with the credentials, for example, to perform a mechanism-specific credential delegation algorithm or set credential attributes according to authorization policy.

Each type of service can play a vital role in the secure middleware infrastructure. Since relying parties must establish trust in an online credential authority, this type of service is more appropriate for long-lived, centrally administered credential management services in environments where establishing trust in additional authorities does not have a high cost. On the other hand, an OCR can be deployed with less overhead by individuals and small groups, and an OCR can be more appropriate for managing credentials in large collaborations, where different sites require different types of credentials and will not trust a common authority.

Two important examples of online credential authorities are Key Distribution Centers (for secret key cryptosystems) and Online Certificate Authorities (for public key infrastructures). Kerberos is a widely adopted implementation of a Key Distribution Center (KDC). In Kerberos, a client sends a request for credentials to the KDC, which returns a credential encrypted with the client's secret key, derived from the user's password. The Kerberos KDC maintains a database of principal names and secret keys. For Online Certificate Authorities (CAs), the client authenticates to the CA and submits a certificate request, and if the client's authenticated identity matches an entry in the online CA's authorization database, the CA returns a signed certificate to the client. The client may authenticate to the online CA with a private key (called an Initial Authentication Key in [1]), a Kerberos credential (as in KX.509 [27] or K5Cert), or some other authentication mechanism.

The IETF Securely Available Credentials (SACRED) working group has published requirements for a credential repository service [3] and is actively developing a credential retrieval protocol. The SACRED group has chosen to implement a repository of opaque credentials by requiring that credentials be opaque to the retrieval protocol and not be present in clear text in the server. Despite this difference with MyProxy, the SACRED effort has contributed a valuable set of requirements for credential retrieval services that will inform future MyProxy work.

The primary risk of any online credential retrieval service is the potential compromise of a large number of credentials. An attacker that compromises an online credential authority can create new security credentials for any identity in the security realm. Similarly, an attacker that compromises an online credential repository has access to all of the credentials stored there. Best practices developed for Kerberos installations can translate directly to methods for maintaining other types of well-secured online credential retrieval services. A professionally administered credential management server should provide a higher level of security than current practice, where end users store their private keys on less secure end systems, including network file systems where eavesdropping on unencrypted network traffic is possible.

Encrypting credentials in the repository, so only the owner can decrypt them, provides additional protection, since the attacker must also break the encryption on each individual credential to compromise it. However, this comes at a cost in convenience, requiring significant functionality to be offloaded to the client. Clients must perform an additional decryption operation, typically requiring an additional pass phrase prompt, after retrieving the credential, failing to meet the single sign-on requirement. To obtain information about stored credentials, the client must retrieve all credentials and decrypt them, rather than sending an authenticated query to the repository service. Finally, storing encrypted credentials makes it impossible to support mechanism-specific delegation protocols, which avoid transferring private keys over the network.

Community authorization [37] enables resource providers to delegate authorization decisions to a community of users. Rather than authorizing each user in a community, the resource provider grants community-wide access to resources, and the community manages its own membership via a Community Authorization Service (CAS), which controls which individuals can access the resources on behalf of the community. Users authenticate to the CAS with their individual identity credential, which may have been retrieved from a MyProxy server, and then retrieve a community identity credential to access community-owned resources. Novotny, Tuecke, and Welch have implemented a prototype CAS as an online certificate authority.

In some scenarios, it may be preferable to implement authorization using attribute certificates [14], which bind attributes such as group membership to identities. The Internet2 Shibboleth project [9] is working toward an attribute-based framework for cross-organizational authentication and authorization. Once issued, attribute certificates can be placed in an online repository, often an LDAP [46] server, for retrieval by relying authorities. Alternatively, attribute certificates can be issued on demand by an online Attribute Authority (AA), which is similar to an online certificate authority: clients authenticate to the AA with their existing set of identity credentials to retrieve attribute certificates.

While there are many types of credential retrieval services, the MyProxy credential repository meets a fundamental need for credential management in the GSI today. Interoperability between different types of credential retrieval services will be addressed through the Global Grid Forum and future work with the Open Grid Services Architecture. This is discussed in more detail in section 2.3.

2 Plan of Work

2.1 Phase 1: NMI Deployment

NMI provides a valuable opportunity to expand MyProxy deployment and gain additional experience with online credential repositories in production middleware environments. Improvements in packaging, documentation, and software quality are required to meet the standard of quality set for the NMI middleware distribution.

The Grid Packaging Technology (GPT) is the mechanism for packaging software for the NMI Middleware distribution, so a first step to preparing MyProxy for inclusion in NMI is to package MyProxy with GPT. MyProxy is currently distributed in source code form only, with a set of simple configuration and installation scripts. Packaging MyProxy with GPT will require identifying dependencies on other GPT packages and writing GPT-compatible build, configuration, and installation scripts.

Current MyProxy documentation is limited to a short introduction for system administrators, a description of each command, and simple installation instructions. Additional documentation will be developed to provide an overview of MyProxy for end-users with limited knowledge of Grid security, instructions for using MyProxy for common tasks, and information on troubleshooting common problems.

Error handling in MyProxy is currently limited, resulting in cryptic output messages that can sometimes confuse users and make it difficult to diagnose problems. Error handling will be improved to provide helpful output for common errors. A set of test cases will be developed to verify correct functionality and appropriate error handling in future releases. With each new release, test cases will be added to verify new functionality and existing tests will be re-run to verify that existing functionality has not been effected.

The MyProxy software modules were developed for internal use within the MyProxy software package. Functionality in these modules will be exposed in a documented Application Programming Interface (API) for use by Grid service developers to integrate MyProxy services, such as automatic credential renewal, into their software. The exposed modules will be reviewed and modified as necessary to ensure they are thread-safe, since many Grid services that need to interface with MyProxy are multi-threaded.

2.2 Phase 2: Near-term Research and Development

2.2.1 Support additional authentication mechanisms to MyProxy

One of the goals of the Grid Security Infrastructure (GSI) is single sign-on, where a user must enter his or her pass phrase only once at the start of a session, rather than re-authenticating for each operation during the session. In practice, however, two sign-on operations are still required. The user first signs on to the local security domain, typically by logging in to the computer, and then signs on to the Grid by entering an additional pass phrase to decrypt the long-term credentials or retrieve them from a MyProxy server. Adding support in MyProxy for local security mechanisms can remove the second sign on, allowing locally authenticated users to use their local credentials to obtain a GSI credential.

This project will modify the MyProxy protocols to provide flexible support for different authentication mechanisms, using the Simple Authentication and Security Layer (SASL)

protocol [30]. Kerberos [34] will be the first additional mechanism to be supported, given its popularity in universities and laboratories in the U.S. Grids today are largely dominated by PKI and Kerberos authentication mechanisms. SASL is chosen so that new authentication mechanisms can emerge, be ported to SASL by the community, and subsequently be immediately usable by MyProxy.

A typical usage scenario facilitated by this new functionality illustrates its value. A site such as NCSA can run one or more MyProxy servers, into which users' long-lived credentials are uploaded. Each individual GSI credential might have a lifetime of 1 year and can either be uploaded by an end-user or by a system administrator, depending on the site requirements. That is, an "expert-level" user might not want a system administrator to upload her credential without her knowing, while "novices" might want the system administrator to do this behind the scenes so that they never have to deal with "security ugliness"! Then, for the next year, every time the user logs onto a local machine, she transparently acquires a valid short-term GSI credential from the MyProxy server(s). In this scenario, this additional functionality in MyProxy both makes the Grid more secure and easier to use by taking the management of GSI keys out of the hands of novice users and by removing the need for multiple pass phrases.

Online credential retrieval services, such as MyProxy, perform a type of credential translation. In the traditional case, the service translates a password (a simple type of credential) into a Kerberos ticket (in the case of the Kerberos KDC) or a PKI certificate (in the case of an online certificate authority or a certificate repository). By adding support for Kerberos authentication, MyProxy provides a service similar to K.X509 or K5Cert, which are online certificate authorities that accept Kerberos authentication. As a credential repository, MyProxy differs from K.X509 and K5Cert in that it does not require relying parties to trust an additional certificate authority but instead stores certificates signed by existing certificate authorities in the PKI.

2.2.2 Support auditing

We will add support for real-time auditing of the MyProxy service. Current auditing support in MyProxy is limited to writing to *syslog* minimal information such as IP address of connector, time of day, and name and lifetime of credential being requested. All operations on the MyProxy server, including credential upload, modification, and retrieval, will be securely logged. Both users and administrators will have secure access to the logs (Currently, only the person with root access on the machine on which MyProxy executes can see the logs). Users will be able to query the logs for transactions involving their credentials—for example, to find out how a credential has been used in the last week. Administrators will be able to query the logs for troubleshooting, attack detection, and attack response. For example, if a credential has been compromised, the MyProxy logs can provide general information about where that credential was used. The interfaces to the audit trail will be designed with the needs of both novice users and advanced system administrators in mind. The auditing functionality will include support for event notification, so users can register for notification when their credentials are accessed and administrators can register for notification of problematic events on the server (for example, a large number of unsuccessful authentication requests may indicate an attack in progress on the server). The first delivery mechanism will be via email; additional delivery mechanisms will be created if situations arise in which email is deemed too slow.

Delegation tracing is closely related to auditing and will be considered in the design of the auditing system. Delegation tracing provides information about the ancestry of a credential, allowing a relying party to make authorization decisions based on how the credential was created or accessed. Delegation tracing also can provide important information when a credential is compromised about where the point of compromise may have been. X.509

certificate extensions for tracing delegation of proxy certificates have been proposed in the Global Grid Forum [41]. However, as credential retrieval systems allow access to credentials via different authentication mechanisms, a mechanism that traces the access history of a credential across mechanisms will be needed.

2.2.3 Improve authorization and access control

The MyProxy server currently supports a limited form of access control on the credentials it manages. A set of server-wide policy expressions state who may store and retrieve credentials from the MyProxy server. This server-wide, *all-or-nothing* approach makes it difficult to use a single server for different scenarios and makes the server administrator responsible for determining the trustworthiness of services that request credentials. Development is underway to allow users to specify a similar set of policy expressions on their credentials to help solve this problem. This allows the server administrator to set a reasonable default access policy, with the knowledge that each user will control how each credential should be used within that policy. For example, the server may allow credential retrieval from any host on the network, but each user can specify the specific hosts from which he or she will be retrieving credentials. Likewise, credential renewal can be enabled on a per-credential basis for specific services, rather than for all credentials on the server or none. Authorization in MyProxy will also be extended to address security for credential renewal by adding restrictions such as a maximum renewable lifetime for a credential or a maximum number of allowed renewals.

The scope of work in this part of the project is both on mechanism and policy. Mechanism, in this case, addresses *how* the authorization checks actually take place. Policy addresses under what conditions a new or refreshed credential is issued. Because of our emphasis on standards-based approaches, we propose to use an existing middleware authorization mechanism such as the Generic Authorization and Access Control (GAA) API [15]. In this way, we will both leverage off of existing work and contribute to the standardization process by identifying new requirements. The identification of reasonable policies for constraining credential renewal and issuance is a longer-term research goal that must be addressed, in part, through experience we will gain in continued deployment of MyProxy. That is, it is appropriate for a user to configure the MyProxy server to only issue his credential to certain physical locations (i.e., IP Addresses), but it is not clear to what degree the *intent* of the request can be captured in the MyProxy protocols and reflected in the authorization policies. This is directly related to the policy issues of GSI Proxy Certificate restrictions: how does a user know precisely what operations to restrict a service from performing without knowing precisely how an operation will perform its task? As users move from the limited model of asking a service to “write file X on site Y for me” toward more abstract measures of “Do whatever it takes to get my task done”, a user cannot wholly anticipate the rights needed to perform an action on his behalf, thus greatly restricting the ability of the user to limit the scope of vulnerabilities through restricted delegation [40]. In building the support for more restricted, user-centric authorization mechanism, we will determine and recommend appropriate authorization policies to Grid practitioners.

2.2.4 Improve support for multiple credentials within MyProxy

As the Grid Security Infrastructure matures, it is expected that a user will need to manage multiple credentials to perform required tasks on the Grid. A user may hold multiple credentials because the different systems he or she accesses have differing policies on the credentials they are willing to accept—for example, they trust mutually exclusive sets of Certificate Authorities. A user may also have access to multiple group or community credentials that enable access to community owned or allocated resources. Furthermore, multiple credentials can help a user

limit vulnerability, by allocating different credentials to different tasks, with access rights specific to each task, so if one credential is compromised, only it must be revoked: the other tasks can continue using their credentials. Finally, a user may have different credentials for different uses: one for authentication, another for contract signing, a third for e-mail encryption, and a fourth for encryption of tape backups. In short, while Section 2.2.1 addresses the requirement for new support for multiple ways in which to authenticate *to* MyProxy, this section addresses the need for the MyProxy server to manage both multiple instances of a particular type of credential.

Currently, MyProxy associates a simple identifying tag with each credential (i.e., the "username"). A user can upload multiple credentials with different tags but she must remember the tags and the characteristics of the associated credentials. MyProxy will be improved to allow users to query the attributes of their stored credentials. The contents of a MyProxy server will be redesigned to be represented as a database to ensure correct and efficient queries (currently, the properties of managed credentials are not treated as first-class entities). Examples of useful query attributes include who issued the credential (i.e., the name of the Certificate Authority), when the credential will expire, and other credential restrictions. The interface should allow a client to query MyProxy to find the appropriate credential for a given task. For example, to authenticate a user to a secure service, the client could first obtain information about what credentials the service is willing to accept, then query MyProxy to find a matching credential to use, if available.

MyProxy will also be modified to store Kerberos credentials, in addition to GSI credentials, to support access to Grid resources in both GSI and Kerberos security domains. Users will be able to upload Kerberos tickets to the MyProxy repository to be retrieved using different authentication mechanisms. Storing both Kerberos and GSI credentials in a single repository will make it more convenient to launch long-running computations across Grids that use GSI and Kerberos, providing a single interface to credential retrieval and renewal for those computations.

2.3 Phase 3: Standardization and OGSA

A critical aspect of this project is the continued standardization efforts, which we are just commencing in the Global Grid Forum (GGF). Basney and Humphrey are both active in the Grid Security Infrastructure (GSI) working group of the Global Grid Forum (GGF). Basney's draft requirements document for online credential retrieval services [5] was well received at the GGF meeting in Toronto in February 2002, and members of the GSI working group indicated interest in continued work in this area to define a standard protocol for retrieving credentials from online credential authorities and repositories.

As part of this project, we intend to support the recently proposed Open Grid Services Architecture (OGSA) [18], a developing standard in the Global Grid Forum based on the Web Services activity of the World Wide Web Consortium (W3C). OGSA builds on standard mechanisms for service description, discovery, and binding to multiple protocols. While OGSA is continuing to evolve, currently, the key aspects that are relevant to MyProxy are that “Grid Services” should actively publish their interfaces (via WSDL [12]), there should be local/remote transparency between clients and servers, servers should support SOAP [6] or some other XML protocol [2], and there should be “standard mechanisms for creating and discovering transient Grid service instances”. As part of this project, MyProxy will be OGSA-compliant, particularly in terms of support for SOAP/XML and publishing the MyProxy interfaces and existence via WSDL and UDDI [42]. In addition, it should be noted that the observed community intent to converge on OGSA strongly supports the need for a service such as MyProxy: transient Grid services will inevitably need to obtain and refresh their credentials from an OCR such as MyProxy. That is,

there will be situations in which a service that a particular user has defined will need to be instantiated, but that user is not currently "present" in the Grid. In this case, some "super" service could be directly empowered to create new services on behalf of the (not present) user, and generate and endow the new service with a valid credential. This could all be performed without an OCR. However, most users will object to the lack of control over their identity and credentials suggested by this model. Instead, MyProxy is the functionality needed by users to more closely control *who* and *what* can operate on their behalf. The observation is that individual users will increasingly define and publish their own services in the OGSA model, utilizing a rich access control mechanism to control who can use their services. Without the use of MyProxy, these users might be less apt to contribute their services if these services can be instantiated and executed seemingly without their knowledge. MyProxy creates a single, auditable "choke point" for this new user-centric model of Grid computing.

Our support for OGSA will build on our experiences determining the requirements of MyProxy for the Legion-G system, which is a new project at UVa to provide the Legion [23] "Look-and-feel" over Globus. In Legion-G, every Grid entity (human, machine, software artifact, etc.) is encapsulated as an object, with its own identity and running on behalf of some user. Legion-G objects are similar to OGSA Grid Services in that they can be temporary and instantiated without the owner being present. Legion-G is not currently integrated with MyProxy; part of the proposed work for this grant is to build the necessary functionality that will support the requirements of new models such as OGSA and more established Grid computing models such as Legion and Legion-G.

Because OGSA has only recently been introduced to the Grid community at the time of this writing, it is not clear to what extent MyProxy will need to be modified to support OGSA compliance. However, it appears certain that the importance of MyProxy in Grid Computing will increase due to OGSA, if only because it is unlikely that all Web Services will be running with GSI credentials. MyProxy will serve as the brokering agent that provides the mechanism by which to translate between Grid security mechanisms.

3 Schedule and Milestones

3.1 Phase 1: NMI Deployment (3 months)

NCSA and UVa will test, harden, document, and package MyProxy for inclusion in the NMI Middleware distribution.

Results: A robust online credential repository for GSI credentials included in the NMI middleware distribution.

Evaluation: Feedback from NMI users as to the utility and quality of the MyProxy software will be solicited to evaluate Phase 1 efforts.

3.2 Phase 2: Near-term Research and Development (9 months)

NCSA will:

- Implement per-credential access control and support for multiple credentials per user.
- Deliver a thread-safe MyProxy API and work with the Condor Team at the University of Wisconsin to use that API to integrate MyProxy credential management into the Condor-G job manager.

- Add support for Kerberos authentication and storing Kerberos credentials in MyProxy.

UVa will implement sophisticated auditing in MyProxy, including an audit log query interface and a notification system for audit events.

Results: A new MyProxy distribution for NMI that supports Kerberos authentication, per-credential access control, multiple credentials, and sophisticated auditing.

Evaluation: The level of adoption of the new MyProxy features among NMI users will be used to evaluate the success of Phase 2. Information about how MyProxy is being used will be solicited and results will be documented.

3.3 Phase 3: Standardization and OGSA (1 year)

NCSA and UVa will collaborate on middleware security research using the MyProxy platform and continue efforts to standardize an online credential retrieval protocol in the Global Grid Forum. Issues of MyProxy availability and scalability will be directly addressed in this phase.

Results:

- An OGSA-compliant protocol draft for credential retrieval protocols submitted to the Global Grid Forum (GGF). NCSA will lead development of an implementation of the proposed protocol.
- A framework for authorization in credential management services. UVa will lead this effort, which will include an implementation of the proposed framework in MyProxy.
- Mechanisms for managing multiple credentials. NCSA and UVa will build on MyProxy support for multiple credentials, implemented in Phase 2, to make working with multiple credentials more transparent. The mechanisms will enable entities to select the appropriate credential for a given task. A negotiation mechanism will allow an entity to choose a credential that its peer will accept. A credential management interface will enable users to more easily choose the appropriate credential to delegate to a task, by granting the rights specifically required for that task in the chosen credential.

Evaluation: The level of adoption of the proposed GGF-standard protocol by other developers of credential retrieval services will be one measure of success. Participation from early adopters will be solicited to evaluate mechanisms for management of multiple credentials and credential authorization.

4 Broader Impact of Proposed Research

The work described in this proposal aims to reduce one of the primary barriers to performing scientific research using middleware technologies, namely, the credential management overhead. A credential repository service enables researchers to focus on their scientific goals, rather than requiring researchers to learn the details of the middleware security infrastructure.

Transparent credential management can make it feasible to use middleware technologies in a classroom setting. For example, the NCSA ChemViz project [11], which gives high school students access to computational chemistry tools via the Web, is currently working to integrate middleware technology into its computing infrastructure. Issuing public key credentials directly to thousands of high school students is not a feasible option. A solution currently under investigation is to store the students' credentials in a MyProxy repository for convenient retrieval by the students when they access the ChemViz Web portal.

This proposal requests funding for graduate student research assistants at NCSA (University of Illinois) and the University of Virginia to participate in research and development in the area of middleware security, thereby supporting graduate research and education in computer science at both institutions.

The results of this project will be disseminated broadly in the form of a freely available source code software distribution, a set of Global Grid Forum draft documents, and one or more technical research papers.

5 Summary

The current approach for authentication and authorization relies too heavily on short-lived credentials, users submitting and monitoring jobs only from their workstations (or some other machine on which they have logged in), and hand-management of multiple types of credentials by end-users. An on-line credential repository rectifies these problems, and in doing so facilitates a *persistent* infrastructure that is more secure because users can rely on a trusted, third-party solution instead of having to create a (potentially insecure) mechanism by themselves. In addition, an on-line credential repository makes a national and international middleware infrastructure more usable because users no longer have to determine which credentials to present to end resources, nor are users in effect restricted to operations only as they sit next to their terminal. As this middleware is deployed, these capabilities will only increase in importance.

The key aspects of this proposal are that MyProxy is an on-line credential repository that has been effectively prototyped and evaluated, representing a design and software maturity necessary for immediate large-scale deployment and impact. In the two years of this project, additional necessary functionality will be created and subsequently redeployed in the NMI middleware distribution, ready for multinational computing projects. PI Basney and co-PI Humphrey are very active in the Grid community and are uniquely qualified to ensure a design and implementation that is suitable for the broad spectrum of underlying technologies and usage models. The Global Grid Forum will both be used as a forum by which to determine a consensus on requirements and also as a venue by which to perform outreach to emerging user communities, thus ensuring broad impact.

References

[1] Adams, C. and S. Ferrell. Internet X.509 Public Key Infrastructure Certificate Management Protocols. IETF RFC 2510, March 1999.

[2] Apparao, V. et al. XML Protocol (XMLP) Requirements. W3C Working Draft 19 March 2001. http://www.w3.org/TR/xmlp-reqs/

[3] Arsenault, A. and S. Ferrell. Securely Available Credentials - Requirements. IETF RFC 3157 (Informational), August 2001.

[4] Baru, C., R. Moore, A. Rajasekar, and M. Wan. The SDSC Storage Resource Broker. In *Proceedings of CASCON'98*, Toronto, CA, 1998.

[5] Basney, J. GSI Online Credential Retrieval - Requirements. Global Grid Forum GWD-I (Informational). February 2002.

[6] Box, D, et al. Simple Object Access Protocol (SOAP) 1.1. W3C Note 08 May 2000. http://www.w3.org/TR/SOAP

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

[8] Butler, R. and T. Genovese. Global Grid Forum Certificate Policy. Global Grid Forum Draft. September 2001.

[9] Carmody, S. Shibboleth Overview and Requirements. Internet2 Draft <draft-internet2-shibboleth-requirements-01.html>, February 2001.

[10] Chapin, S. J., D. Katramatos, J. Karpovich, and A. Grimshaw. Resource Management in Legion. *Future Generation Computer Systems*, Vol. 15, pp. 583-594, 1999.

[11] ChemViz: Chemistry Visualization Program at NCSA. http://chemviz.ncsa.uiuc.edu/

[12] Christensen, E., F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. W3C Note, 15 March 2001. http://www.w3.org/TR/wsdl

[13] Dierks, T. and C. Allen. The TLS Protocol Version 1.0. IETF RFC 2246 (Standards Track), January 1999.

[14] Farrell, S. and R. Housley. An Internet Attribute Certificate Profile for Authorization. Internet draft <draft-ietf-pkix-ac509prof-09.txt>, IETF PKIX Working Group, June 2001.

[15] Ferrari, A., F. Knabe, M. Humphrey, S. Chapin, and A. Grimshaw. A Flexible Security System for Metacomputing Environments. In *Seventh International Conference on High Performance Computing and Networking Europe (HPCN Europe 99)*, pages 370-380, April 1999.

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

[17] Foster, I. and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputing Applications*, 1997.

[18] Foster, I., C. Kesselman, J. Nick, S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. January 2002. http://www.globus.org/ogsa/

[19] Frey, J., T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke. Condor-G: A Computation Management Agent for Multi-Institutional Grids. *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)*, 2001.

[20] Global Grid Forum, http://www.gridforum.org

[21] Grimshaw, A.S. Portable Run-Time Support for Dynamic Object-Oriented Parallel Processing. *ACM Transactions on Computer Systems*, pp. 139—170, Vol. 14, No. 2, May 1996.

[22] Grimshaw, A.S., W. Wulf, and the Legion team. The Legion Vision of a Worldwide Virtual Computer. *Communications of the ACM*, 40:1, pp. 39-45, January 1997.

[23] Grimshaw, A. S, A. Ferrari, F. Knabe, and M. Humphrey. Wide-Area Computing: Resource Sharing on a Large Scale. *Computer*, 32(5):29-37, May 1999.

[24] Humphrey, M. and M. Thompson. Security Implications of Typical Grid Computing Usage Scenarios. *Cluster Computing*, 2002. *To Appear.*

[25] ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, June 1997.

[26] Katramatos, D. M. Humphrey, A. Grimshaw, and S. Chapin. JobQueue: A Computational Grid-wide Queuing System. *Proceedings of the 2nd International Workshop on Grid Computing*, Denver, CO, November 12, 2001.

[27] Kornievskaia, O., P. Honeyman, B. Doster, and K. Coffman. Kerberized Credential Translation: A Solution to Web Access Control. *USENIX Security Symposium*, Washington, D.C., August 2001.

[28] Linn, J. Generic Security Service Application Program Interface Version 2, Update 1. IETF RFC 2743 (Standards Track). January 2000.

[29] Microsoft .NET Passport Single Sign-on Service, http://www.passport.com

[30] Myers, J. Simple Authentication and Security Layer (SASL). IETF RFC 2222 (Standards Track). October 1997.

[31] Natrajan, A., M. Crowley, N. Wilkins-Diehr, M. Humphrey, A. Fox, A. Grimshaw, C. L. Brooks, III. Studying Protein Folding on the Grid: Experiences using CHARMM on NPACI Resources under Legion. *Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, San Francisco, CA, August, 2001.

[32] Natrajan, A, M. Humphrey, and A. Grimshaw. Capacity and Capability Computing using Legion. *Proceedings of the 2001 International Conference on Computational Science*, San Francisco, CA, May 2001.

[33] Nguyen-Tuong A., *et al.*. Exploiting Data-Flow for Fault-Tolerance in a Wide-Area Parallel System. *Proceedings of the 15th International Symposium on Reliable and Distributed Systems (SRDS-15)*, pp. 2-11, 1996.

[34] Neuman, B. C. and T. Ts'o. Kerberos: An Authentication Service for Computer Networks. *IEEE Communications*, 32(9):33-38. September 1994.

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

[36] Novotny, J., S. Tuecke, and V. Welch. An Online Credential Repository for the Grid: MyProxy. *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press, August 2001.

[37] Pearlman, L., V. Welch, I. Foster, and C. Kesselman. A Community Authorization Service for Group Collaboration. Draft submitted to *2002 IEEE Worksop on Policies for Distributed Systems and Networks*.

[38] Portable Batch System, PBS. http://www.openpbs.org

[39] Ryutov, T., G. Gheorghiu, and C. Neuman. An Authorization Framework for Metacomputing Applications. *Cluster Computing*, 2(1999), pp. 165-175.

[40] Stoker, G., B. White. E. Stackpole, T.J. Highley, and M. Humphrey. Toward Realizable Restricted Delegation in Computational Grids. In *Proceedings of the International Conference on High Performance Computing and Networking Europe (HPCN Europe 2001),* Amsterdam, Netherlands, June 2001.

[41] Tuecke, S., D. Engert, I. Foster, L. Pearlman, and C. Kesselman. Internet X.509 Public Key Infrastructure Proxy Certificate Profile. *Draft-ietf-pkix-proxy-01.txt*, Draft of August 2001.

[42] Universal Description, Discovery and Integration, UDDI. http://www.uddi.org/

[43] White, B., A. Grimshaw, A. Nguyen-Tuong. Grid-Based File Access: The Legion I/O Model. In *Proceedings of the Ninth IEEE International Symposium on High Performance Distributed Computing*, Pittsburgh, PA, August 2000.

[44] White, B., M. Walker, M. Humphrey, and A. Grimshaw. LegionFS: A Secure and Scalable File System Supporting Cross-Domain High-Performance Applications. In *Proceedings of Supercomputing 2001*, Denver, CO, November 2001. *Best Student Paper Finalist.*

[45] Wolski, Rich. Dynamically Forecasting Network Performance using the Network Weather Service. *Cluster Computing*, 1(1):119-132, 1998.

[46] Yeong, Y, T Howes, and S. Kille, Lightweight Directory Access Protocol. IETF RFC 2251 (Standards Track), December 1997.