FINE-GRAINED AUTHORIZATION IN THE GREAT PLAINS NETWORK VIRTUAL ORGANIZATION

A Thesis

Presented to the Faculty of the Graduate School

University of Missouri-Columbia

In Partial Fulfillment of the Requirements for the Degree

Master of Science

By IONUT OVIDIU CIORDAS

Dr. Gordon K. Springer, Thesis Supervisor

AUGUST 2007

The undersigned, appointed by the Dean of the Graduate School, have examined the thesis entitled

FINE-GRAINED AUTHORIZATION IN THE GREAT PLAINS NETWORK VIRTUAL ORGANIZATION

Presented by

Ionut Ovidiu Ciordas

A candidate for the degree of

Master of Science

And hereby certify that in their opinion it is worthy of acceptance.

_____

Dr. Gordon K. Springer

_____

Dr. Wenjun Zeng

_____

Dr. William H. Miller

ACKNOWLEDGEMENTS

The creation of this thesis has been a long and rewarding journey that couldn't have been possible without the wonderful people that lent their time, expertise and support to help me achieve it. I want to express my sincere gratitude and appreciation for everyone who has been supportive in many different ways.

Firstly, I want to thank my thesis advisor and professor, Dr. Gordon K. Springer. His patience and guidance in leading me on the right path of identifying and describing the issues that this thesis deals with, his constructive criticism and insight have helped me grow professionally and be more confident in my abilities, this thesis included. Without his valuable knowledge, time and patience, the path that I followed in order to write this thesis would have been deterred too many times.

Secondly, I want to thank Dr. William Miller and Dr. Wenjun Zeng for taking time from their extremely busy schedules to serve on my thesis committee and contribute valuable insights.

Lastly, I also want to express my gratitude to my parents Vasile and Eugenia for their unabridged love and support for all my decisions even if they were sometimes hard to accept. Also I want to thank my amazing wife-to-be Rachel whose love, support and backing have helped me tremendously through this journey.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

FINE-GRAINED AUTHORIZATION IN THE GREAT PLAINS NETWORK VIRTUAL ORGANIZATION

Ionut Ovidiu Ciordas
Dr. Gordon K. Springer, Thesis Advisor

ABSTRACT

The last few years have experienced a steady growth in research institutions showing interest in developing research projects that involve more than one institution's computing resources forming so called virtual organizations. The goal of any virtual organization is to provide member institutions with a free of inter-operability problems, easy to use, secure and robust collaborative research environment.

Shibboleth was one of the choices of infrastructure to be used to create a collaborative inter-institutional research environment as it provides the possibility to share resources across multiple institutions without having to create physical user login ids on each of the systems that are shared. The creation of a collaborative research environment that spans multiple institutions each with their own policies and conventions is a real challenge. In the standard Shibboleth architecture, the identity provider (the user's home institution) is in charge of authenticating the user and also of storing all the entitlements that the service provider (the shared resource) is basing their access control decisions. Business and user privacy policies make it difficult to deal with the storage and management of all the entitlements in the identity provider. One problem pertains to the security risks involved in allowing external entities to manage entitlements in the identity management system of an institution. Another problem emerges from the identity providers' usage of the eduPerson object class to store the entitlements that belong to a user. The eduPerson object class does not support significant modifications in its format to allow for virtual organization-defined entitlements to be stored in its structure.

This thesis addresses some of the issues that have slowed the adoption of Shibboleth-based authentication and authorization systems in deploying fully collaborative research environments. In order to allow for fine-grained authorization at the virtual organization level, there is a need to define, manage and use virtual organization entitlements independently of any institution or participating company. These entitlements encompass the idea of a shared resource that needs to be made available to any entitled entity from any member organization. In order to attain this goal, the definition, management, and usage of the virtual organization-defined entitlements are maintained separately from the identity provider – the entitlements repository. The identity provider is in charge of authenticating the users, the service provider is in charge of the authorization decisions, and the entitlements repository is in charge of defining, managing and providing access for queries and updates to be run by the stored entitlements service. The identity provider, the service provider and the entitlements service jointly provide for creating a secure and robust collaboration environment for use by virtual organizations.

1. Introduction

The need for resource sharing in the higher education research community has increased significantly over the last few years. Researchers at top universities have started to harness the communication and collaboration power of the internet expansion. Groups from multiple universities and companies have formed to work on projects of common interest. In this process, they started sharing computing resources that belong to one or more of the respective institutions. In order to easily, securely, and transparently use a computing resource belonging to another institution, each participant needs to obtain a user name and a password from the resource's administrators in order to get access to the resource or service.

In order for these groups of researchers to have a meaningful and efficient experience using their institutions' common computing resources, a framework needs to be established that can govern the relationships between the participating institutions. The policies and the trust that they bring to the shared research environment is called a virtual organization or a federation of institutions. An institution can be participating in more than one virtual organization, establishing a set of rules and policies with each one of the virtual organizations that it joins.

The Great Plains Network (GPN) is a federation of research universities that has encountered the particular problem of protecting computing resources from abusive use, while allowing other members of the federation to use them without creating and maintaining a plethora of authentication credentials (e.g., user names and passwords).

GPN chose Shibboleth as the middleware architecture to support the distributed authentication and authorization scheme that is needed in order to achieve controlled access for authorized users at shared computing resources. Shibboleth is a middleware initiative from Internet2/MACE (Middleware Architecture Committee for Education) that addresses this problem by efficiently offering a mechanism to authenticate and authorize inter-institutional user access to protected resources [MACE].

Shibboleth creates a mesh of trust relationships among the federation's institutions based on public key certificates and agreed-upon policies. The Shibboleth mechanism allows for a user that has been authenticated by his or her home institution to access and try to use resources available to any virtual organization that his or her institution is a member. Upon successful authentication, the user is able to access all the resources that any virtual organization encompasses, subject to an access policy established by each participating institution.

In general, a user that requests access to a protected shared computing resource firstly needs to be authenticated by his or her home institution. Only after the user is properly authenticated, the requested resource authorizes the user to use it or deny access based on an access control policy established by the institution that owns the resource. The natural order of authentication followed by authorization is also followed by Shibboleth. In this context, an identity provider represents the Shibboleth entity that authenticates a user and answers attributes inquiries from the service provider. The identity provider is embedded in an institution's identity management system – the system that holds records of all the students, faculty and staff that belong to a university and makes authentication decisions

based on the presented credentials. A service provider represents the Shibboleth entity that communicates with the user, the user's identity provider at his or her home institution, and makes the access control decision based on the user's entitlements. An entitlement is a piece of data that allows a user access to a specific resource, service or a group of computing resources described by the entitlement's value. An attribute is an atom of information about a user, such as "enrolled in CS 1010" or "member of the ABC project." The entitlements that are part of the user's identity management record need to be stored by the home institution's identity management system and presented upon request to the service provider. In order to authorize someone to use a computing resource, the service provider entity needs to establish if a user has the right attributes and entitlements in order to be allowed to use the specific resource or service.

The Shibboleth mechanism offers a secure and fast way of exchanging the attributes in order to authorize a potential user to a computing resource. A practical issue of user privacy and institutional business and privacy practices has delayed the wide adoption of the system. Institutions are reluctant to allow people from outside institutions to have restricted or unrestricted access to an institution's identity management directory in order to administer the access control entitlements. As such, this thesis presents an approach and an implementation of a service that offers storage and query capabilities to service providers across a federation. Each service provider that belongs to any one or more federations of institutions can use this service in order to decide if a user who requests access to a protected computing resource has the necessary access entitlements. This approach allows for the decoupling of the identity provider's authentication main mission and the service provider's authorization capabilities. Using the proposed service, the

identity providers do not have to deal with storing entitlements locally in their identity management systems and the service providers do not have to worry that they are not able to define, describe and administer their own users' entitlements.

The proposed service provides user data privacy and protection mechanisms allowing for extensive user data confidentiality. As such, all the user's entitlements pertaining to one or more virtual organizations' resources can be stored securely in one central entitlements database managed by the proposed entitlements management service. Each service provider has access to modify and query the service about user entitlements while the entitlements server is located somewhere in the Internet. Based on the received information, the service provider can reach an access control decision. The system allows designated administrators from the federation's member institutions to have access and update the server's database in order to maintain an accurate image of which users are allowed to use the available resources.

The proposed service's integration with the Shibboleth environment is described in Figure 1. There are n users who want to be granted access to n resources, each user wants access to the same resource as described by its identifier. Firstly, each user authenticates at his or her home institution's identity provider – in this case Idp1 through IdP n. Secondly, the service providers for the requested resources start gathering the information needed to make an access control decision for each user. For instance, SP 2 asks IdP 2 questions pertaining to the user's home institutional data (e.g., "Is user 2 a student?" or "Is user 2 enrolled in CS 1001?"). Once this information is gathered, the service provider

asks all the inter-institutional or virtual organization entitlements questions to the Entitlement Server (e.g., "Does the user have the "urn:mace:greatplains.net:biogrid" entitlement?"). A virtual organization's computing resources span multiple institutions and the entitlement needed to access any one or group of resources do not concern any one member institution. The IdPs from each institution need not know these inter-institutional pieces of information. This is the type of entitlement information that is stored in the proposed entitlements server.

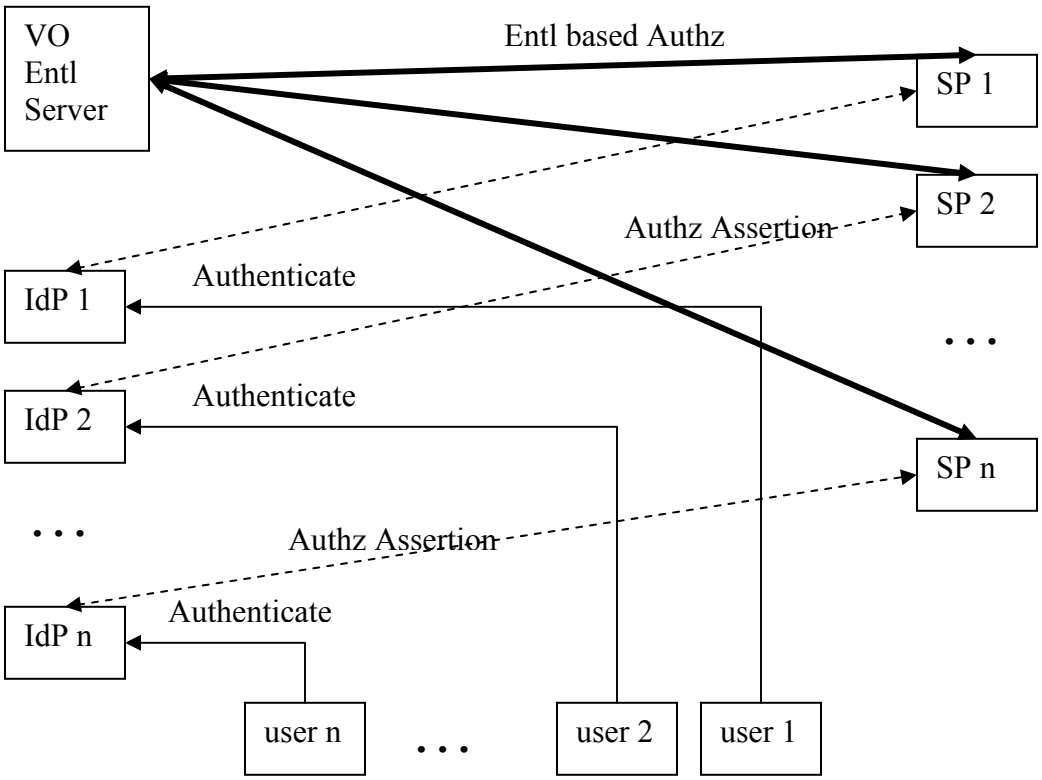

Figure 1. Proposed fine-grained authorization for any virtual organization

An entitlement stored in the entitlements database may span multiple computing resources located at multiple locations belonging to different institutional members of

any virtual organization. For instance, an entitlement with the value "urn:mace:greatplains.net:biosci" grants access to all the resources of the GPN federation. Whoever has this entitlement is able to access all the shared resources of GPN. The access control decision for a particular machine is reached by the service provider which decides based on the entitlements' values passed to the proposed entitlements server.

2. Background

2.1 What is Shibboleth?

Shibboleth is a project of the Internet2/MACE (Internet 2's Middleware Architecture Committee for Education) consortium that is developing a comprehensive framework to support inter-institutional resource sharing subject to access control mechanisms [SHIB]. Shibboleth is comprised of architectures, policies, protocols, technologies and an open-source implementation. Shibboleth is a protocol and architecture for sharing user attributes among institutions that have an established trust relationship in order to form a virtual organization. Its main objective is to facilitate the authentication followed by the authorization of a user to use a shared web resource at a different institution than the user's home institution using the user's home institution credentials and attributes.

The word Shibboleth has historical significance with the topics of security and authentication apart from the information technology perspective. "There was a word which was made the criterion by which to distinguish the Ephraimites from the Gileadites. The Ephraimites, not being able to pronounce "sh", called the word shibboleth."[1] According to the Bible, two Semitic tribes - the Ephraimites and the Gileadites – were engaged in a great battle. The Gileadites had defeated the Ephraimites, and set up a blockade to catch the fleeing Ephraimites. Their sentinel asked each person to pronounce the word "shibboleth". But because the Ephraimites had no "sh" sound in their language, they pronounced the word as an "s" and were thereby revealed and slaughtered. Accordingly, any person who doesn't know the pass word is easily

[1] Book of Judges XII

identifiable and excluded from the group. According to the Webster's Dictionary, the word "shibboleth" represents a criterion, a test or watchword of a party or group; a party cry or pet phrase, a secret.

2.2 Key Shibboleth Concepts and Goals

Shibboleth is intended to solve problems of the following nature: a campus or a company has several web-accessible resources that it wants to share with authorized users from research groups from other institutions. For instance, a company wants to provide access to one limited project's infrastructure to its researchers and also some authorized users from other companies that are part of the same interest organization; a library wants to offer differentiated access to its collection of online paid resources to only a few groups of interested researchers in order to keep the costs low or a researcher wants to get access to an information repository without revealing the user's identity, but being authorized to use it nevertheless.

The users of a Shibboleth system do not need to reveal any private information about themselves, above the required level of credentials ownership required to get authenticated by their home institution (e.g., username-password pair, private-public key pair, etc.). Once authenticated by their home institution, users can have access potentially to a large number of resources available to any of the virtual organizations that their home institution is a member. In order to create a federation of institutions, various institutions agree to use a predefined set of policies that regulate their usage of the shared computing or non-computing resources and the gained knowledge. The established

polices set the ground for the use of commonly accepted and trusted certificate authorities in order to be able to know certainly when a message comes from a specific virtual organization member institution. In order for Shibboleth to be able to make access control decisions, each user needs to be related to an identity provider of an organization that is a member of a virtual organization.

Shibboleth is designed to function independently of any particular authentication schemes or identity management systems deployed by each particular organization (e.g., LDAP, Windows Active Directory, etc.). The only thing that all these institutions have to agree upon is to allow the Shibboleth fine-grained authorization to function based on attributes. Each university should be able to provide access to attributes about its users, subject to the user's control. The users' records contain attributes whose values uniquely describe a user's access capabilities or entitlements. Attributes such as "student in CA 1000" or "faculty" can describe a user's status with the university.

Shibboleth is based on the SAML (Security Assertion Markup Language) [SAML] protocol defined by OASIS (the Organization for the Advancement of Structured Information Standards) [SAML] in order to securely communicate users attributes using an implementation-independent protocol based on XML (eXtended Markup Language). SAML is an XML standard for exchanging authentication and authorization data between security domains, e.g., between an identity provider and a service provider. All messages exchanged in a Shibboleth communication are based on SAML. The Shibboleth and SAML design processes have been coupled to insure that Shibboleth is standards-based.

2.3 Authentication and Authorization with Shibboleth

The issues of authentication and authorization are of utmost importance for the presented examples and for any type of resource or computing resource sharing among institutions. A specific aspect of Shibboleth is the separation of the authentication step and the authorization step in an identity establishing transaction.

Authentication is the process of establishing if a user or service is the person or service that they claim to be and it represents the former of the two. This can be done using usernames and passwords, public and private keys or smart cards combined with some type of knowledge possession (e.g., answering a question presumably answerable only by the individual user).

Authorization is the process of determining, after a user is authenticated, if a user is allowed to have access to a particular resource. Authorization always implies the existence of a previous authentication and as such it represents the latter of the two.

The authorization process can be performed using two approaches. One method uses the user's identity information that gets passed to the resource to make authorization decisions based on the identity and also additional attributes. The second approach is based on attributes about the prospective user that get exchanged until the requested resource reaches an authorization decision. The former approach requires the user to trust the resource, in the latter case the user doesn't need to reveal his/her identity. This latter approach is used by Shibboleth in order to protect the user's privacy.

Authentication is very hard to establish in the absence of trust. As such, one of the goals of Shibboleth is to establish an architecture and a policy that leads to the creation of a fabric of trust among all participating institutions of a virtual organization. This fabric of trust is also called a federation or virtual organization, because some institutions agree upon certain policies that govern their future interactions from the point of view of sharing web-enabled resources. The establishment of the trust fabric is central to the architecture of Shibboleth and a major design goal.

2.4 Trust and Privacy with Shibboleth

In order to establish trust among participating institutions two methods are employed by Shibboleth. Collaborative trust is used when two or more institutions agree to cooperate and share their resources based on an attributes-driven access control system. The collaborative trust must be established through negotiations and strong relationships. Many privacy, business and legal policies have to be agreed upon, adopted and used by all the institutions involved. These policies need to completely specify the types of attributes to be used in the authorization process in order to authorize a user, the possible values of these attributes, and the level of security expressed by the used cryptographic technologies. Hierarchical trust networks of certificate authorities are the trust method employed by Shibboleth. During any Shibboleth data exchange, the two communicating institutions ensure that the other party is the one it pretends to be by the use of mutual certificate-based authentication.

By using Shibboleth, the participating institutions agree to be the brokers between the user and the service from the point of view of authentication and authorization. The service providing institution trusts the authentication handle and the attributes sent by the authenticating institution. Also the sending institution trusts the receiver of its attributes to use them as previously agreed upon to determine the right of a user to access a particular resource and not to use them for other purposes or alter them in the process.

As privacy is one of the Shibboleth's most important goals, privacy issues arise from the way the user's credentials are manipulated by the requested resource. The first approach - passive privacy - requires the user to trust the resource and to hope that the resource complies with the government privacy regulations. The second approach - active privacy - is based on attributes about the user that the user passes along to the resource. These attributes do not need to be user identifiable; a simple handle suffices. Also the user has the chance to authorize what attributes are released to what resource. Once the resource receives all the needed attributes it can make a decision upon the user's access request. Shibboleth's design was built with privacy concerns and has adopted the active privacy paradigm allowing only the user specified attributes to be released to a particular resource.

2.5 Protocol and Functionality

Shibboleth is comprised of three main software components: the identity provider (IdP), the service provider (SP) and the Where Are You From (WAYF) server.

The IdP component of Shibboleth is integrated with the user's home institution identity management system providing authentication services for the institution. The IdP authenticates a user using the home institution's authentication mechanisms, creates and sends a unique handle to the service provider in order to uniquely identify the current transaction. The identity provider also sends user attribute information to the requesting service provider in order to support the SP's access control decision. The actual user attributes are supposed to be stored in a local identity management directory.

The service provider is the Shibboleth component that protects a resource against non-authorized user access. It accomplishes this by deciding if a user should be allowed to access its service based on the user's attributes.

The WAYF service is an independent service operated by a particular service provider or globally by a virtual organization – the Great Plains Network is using a test WAYF service in order to be able to have complete control over its virtual organization's deployment and policies. Its purpose is to identify a user's home institution and to redirect the user to the home institution's authentication system.

The Shibboleth protocol is shown in Figure 2 and is comprised of the following steps:

1. A user requests access to a resource's website.
2. The service provider's SHIRE (Shibboleth Indexical Reference Establisher) redirects the user to the WAYF (Where Are You From) service.

3. The WAYF (Where Are You From) service asks the user to identify its home institution from a list of provided universities.
4. The WAYF (Where Are You From) redirects the request to the user's home institution handle service.
5. The user's home institution requests the user's credentials
6. The user's home institution's handle service authenticates the user.
7. The handle service generates a unique handle that is sent to the service provider's SHIRE (Shibboleth Indexical Reference Establisher).
8. The SHIRE passes the handle to the SHAR.
9. The SHAR (Shibboleth Attribute Requester) doesn't know anything about the handle's user attributes. It only knows that the handle comes from a trusted institution. It sends attribute assertions to the identity provider's attribute authority.
10. The attribute authority looks up the requested attributes according to the user's release policy and it sends the required attributes back to the SHAR (Shibboleth Attribute Requester).
11. The SHAR (Shibboleth Attribute Requester) passes the attributes to the resource manager to make an access control decision based on the received attributes.
12. Once a decision is reached, the resource manager sends a decision answer back to the user.

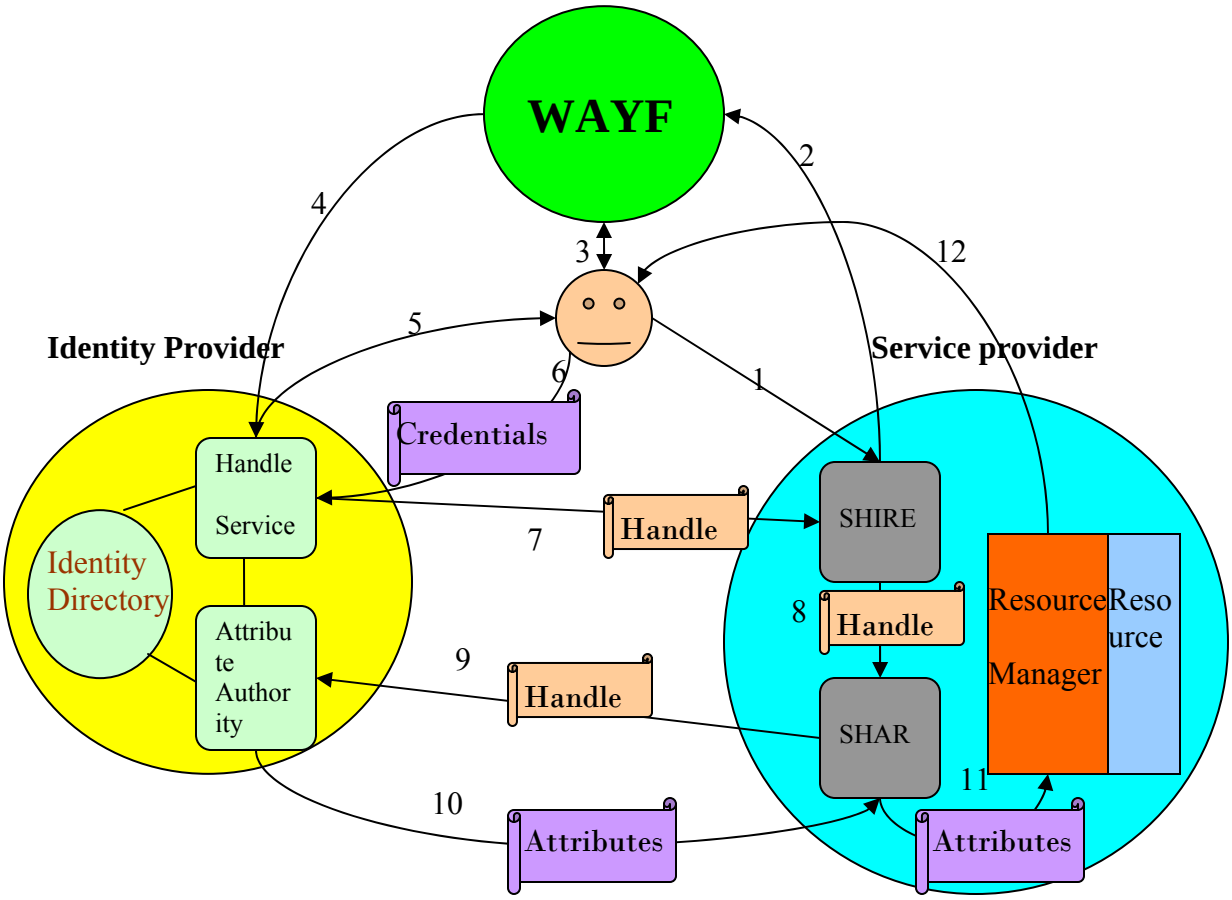

Figure 2. Shibboleth Protocol Outline

Figure 2 is describing the previously outlined steps in a more visual fashion.

- Step 1 is used by the user who requests a web-based shared computing resource by directing their web browser to the internet address of the computing resource at the service provider.
- In Step 2, once the shared resource's service provider receives the user's request for service, the service provider's SHIRE redirects the user's browser to the WAYF service.
- The WAYF service – in Step 3 – communicates with the user and finds out from the user what identity provider the user has an affiliation with.
- In Step 4, the WAYF service redirects the user's browser to the user provided identity provider's handle service.
- Step 5 is used by the identity provider's handle service to ask the user for their credentials and Step 6 is used to make an authentication decision based on the organization's identity management system, if the user has not been already authenticated and a handle already exists for him.
- In Step 7, if the user is authenticated, the identity provider creates a unique handle to identity the user and sends this handle to the service provider's SHIRE which passes it on to the SHAR in Step 8, otherwise the user is presented with a web page that states the identity provider's inability to authenticate the user based on the presented credentials.
- Step 9 is used to send queries about the user's entitlements to the identity provider's attribute authority based on the previously received handle.

- In Step 10, the identity provider's attribute authority sends its entitlements response back to the service provider SHAR.
- After the service provider's SHAR gathers all the needed attributes about the user based on potentially multiple queries to the identity provider's attribute authority, it sends the attributes to the service provider's resource manager that makes the access control decision in Step 11. If there are any other entitlements that are needed in order to reach a decision, the SHAR needs to contact the identity provider's attribute authority and retrieve them.
- In Step 12, the service provider's resource manager sends its authorization decision to the user and in case of an affirmative decision; the user's web page is redirected to the initially requested shared computing resource.

2.6 The Identity Provider Components

The identity provider is comprised of three services: the handle service, the attribute authority and a directory database.

The handle service is designed to accept redirects from the WAYF service and decide if a user has already been authenticated. If the user has not been authenticated, the user is presented with the local institution's authentication procedure. Following a successful authentication, the handle service generates a unique handle to identify the particular user without necessarily disclosing his/her identity to the service provider. The handle service sends the generated handle directly to the service provider's SHIRE (Shibboleth

Indexical Reference Establisher) service. The Handle Service has already received from the WAYF the name of the requested service provider. This information is vital in order to use appropriate encryption technologies with the corresponding public-private keys.

The attribute authority is the service in charge of retrieving user attributes requested by the service provider's SHAR (Shibboleth Attribute Requester). The attribute authority determines which attributes can be released according to the user's release policy. The users have the right to set an authorization policy on their own attributes. Once the attributes are identified, the attribute authority queries the institution's identity management system to actually retrieve the needed values. They are subsequently sent to the service provider's SHAR (Shibboleth Attribute Requester).

It is important to note that some service providers may require identity information (e.g., principal name, principal affiliation) to be provided by the identity provider. The user has control over the release of their information and can allow or disallow the release of data on an assertion request from the service provider. If disallowed, the authorization process fails. Otherwise, if allowed, the authorization process can continue and the service provider can use the identity information to make access control decisions – according to defined policies.

The institution's identity management system is an institutionally managed identity database that holds some of the attributes associated with a user – at least the ones that uniquely identify the user as a member of the institution. It is usually housed by the

institution's LDAP server or its Windows Active Directory service. The directory database receives requests for user attributes from the attribute authority and sends back its findings.

2.7 The Service Provider Components

The service provider is comprised of three services: the SHIRE (Shibboleth Indexical Reference Establisher), the SHAR (Shibboleth Attribute Requester), and the resource manager.

The SHIRE service is designed to receive user requests for its resource and decide if the user has already been authorized to use the particular resource. If the user has not been authorized yet, the SHIRE service redirects the user to the WAYF service in order to identify the home institution and be sent to the home institution's handle service. The SHIRE is in charge of receiving user handles from the handle service on the identity provider side. These handles are sent to the SHAR for further processing.

The SHAR receives handles from the SHIRE and tries to determine the required set of attributes for that user needed to be granted access to its resource. Initially, the SHAR does not know anything about that handle except that it comes from a trusted institution. It sends attribute assertions to the identity provider's attributes authority and waits for the response. Once the attributes are gathered, the SHAR sends them to the resource manager. The SHAR asserts various attributes as a way of describing the need for a user to have specific attributes in order to access the resource.

The resource manager is responsible for making an acceptance or denial decision based on the attributes sent by the SHAR (Shibboleth Attribute Requester). The decision is communicated to the user directly. If the user has been granted the right to access the resource, the user is transferred to the resource.

2.8 The WAYF (Where Are You From) Component

In a virtual organization where many institutions share a trust relationship, the service provider needs to know where to get the handle and the attributes. A special service has been designed to allow the user to select their home institution as a means of getting authenticated and the service provider to receive their handle. The WAYF (Where Are You From) service is usually represented by a web-page with a drop-down list for the user to select their home institution. Once the selection is made, the user gets redirected to their home institution's authentication page.

2.9 Shibboleth and Trust

Shibboleth has been designed with the goal of providing a secure and trustworthy environment for inter-institutional transactions to take place. The virtual organization or federation provides a common set of policies that govern the interactions among the users, their identity provider and the resource's service provider. These polices are the basis for a trust fabric across the entire virtual organization.

The identity provider and the service provider collaborate through the use of technologies (e.g. SAML - Security Assertion Markup Language) and policies to create a collaborative environment that allows for a privacy-preserving context for the Shibboleth users. The identity provider authenticates the users and asserts attributes about the users. The user trusts the identity providers to correctly represent its attributes to the service providers and also to follow strict privacy regulations. The identity provider trusts the service provider not to misuse the user's attributes and to make a correct decision based on those attributes. When an identity provider receives a request for attributes from a service provider's SHAR, the communication needs to take place over a secure communication channel (Secure Socket Layer) and the name from the handle and the name from the certificate need to be part of the same institution and map correctly to the same user.

The service provider requests attributes about the user directly from the identity provider and reaches access control decisions based on those attributes. The service provider trusts the identity provider to accurately communicate and manage user attributes and respect a user's privacy rights. When a service provider receives a handle from an identity provider, the message needs to be signed by the identity provider and the identity provider must be part of the common virtual organization.

Users have the right to authorize the release of their own attributes according to their authorization policies. The same right is reserved for an identity provider in order to be able to securely manage its attributes database and establish corresponding policies.

2.10 Security Assertion Markup Language (SAML) Assertions

SAML is a secure XML (eXtended Markup Language – see Glossary) standard that is built by using the expressiveness power of XML and the security of asymmetrical encryption technologies (e.g., RSA). SAML is used by Shibboleth in order to exchange information between the service provider and the identity provider. The information can take the form of the authentication handle that the identity provider sends to the service provider, the attributes assertions that the service provider sends to the identity provider and also the results of these assertions.

In Shibboleth language, the information stream flowing between the identity provider and the service provider is called SAML assertions. The assertions can contain elements that the service provider can use to make access control decisions for their protected resources. There are three types of SAML assertions: authentication statements, attribute statements and authorization decision statements.

2.10.1 SAML Authentication Statement

The authentication statement comes from the identity provider and asserts to the receiving service provider that the user has been authenticated by the identity provider. The authentication took place at a specific time and using a specific method – all expressed in the authentication assertion. The following authentication assertion contains a <saml:AuthenticationStatement> attribute, as shown in the example below.

```
<saml:Assertion
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion" MajorVersion="1"
  MinorVersion="1" AssertionID="a75adf55-01d7-40cc-929f-dbd8372ebdfc"
   IssueInstant="2007-03-05T09:22:02Z"
```

```
  Issuer="https://idp.example.org/shibboleth">

<saml:Conditions
      NotBefore="2007-03-05T09:17:02Z"
      NotOnOrAfter="2007-03-05T09:27:02Z">

       <saml:AudienceRestrictionCondition>
       <saml:Audience>http://sp.example.org/shibboleth</saml:Audience>
       </saml:AudienceRestrictionCondition>
 </saml:Conditions>

<saml:AuthenticationStatement
      AuthenticationInstant="2007-03-05T09:22:00Z"
      AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">

      <saml:Subject>
            <saml:NameIdentifier
            Format="urn:mace:shibboleth:1.0:nameIdentifier"
             NameQualifier="https://idp.example.org/shibboleth">
              3f7b3dcf-1674-4ecd-92c8-1544f346baf8
            </saml:NameIdentifier>

            <saml:SubjectConfirmation>
            <saml:ConfirmationMethod>
                  urn:oasis:names:tc:SAML:1.0:cm:bearer
            </saml:ConfirmationMethod>
            </saml:SubjectConfirmation>
      </saml:Subject>
 </saml:AuthenticationStatement>
 </saml:Assertion>
```

The authentication statement contains information about the time and the method of authentication used by the user's identity provider. Under the <saml:NameIdentifier> element, the authentication statement contains the unique session handle that has been generated by the identity provider's handle service.

2.10.2 SAML Attribute Statement

The SAML attribute statement is used by the service provider to query the identity provider for values of various attributes about the user to be authorized. The values of these attributes are used in the access control decision making process. An example follows:

```
<saml:Assertion
      xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
      MajorVersion="1" MinorVersion="1"
      AssertionID="a144e8f3-adad-594a-9649-924517abe933"
      IssueInstant="2007-03-05T09:22:05Z"
      Issuer="https://idp.example.org/shibboleth">

      <saml:Conditions
            NotBefore="2007-03-05T09:17:05Z"
            NotOnOrAfter="2007-03-05T09:52:05Z">

      <saml:AudienceRestrictionCondition>
      <saml:Audience>http://sp.example.org/shibboleth</saml:Audience>
      </saml:AudienceRestrictionCondition>
      </saml:Conditions>

       <saml:AttributeStatement>
            <saml:Subject>
            <saml:NameIdentifier
                  Format="urn:mace:shibboleth:1.0:nameIdentifier"
                  NameQualifier="https://idp.example.org/shibboleth">
             3f7b3dcf-1674-4ecd-92c8-1544f346baf8
             </saml:NameIdentifier>
             </saml:Subject>

       <saml:Attribute
             AttributeName="urn:mace:dir:attribute-
      def:eduPersonPrincipalName"
      AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:ur
i">
       <saml:AttributeValue Scope="example.org">
             userid
       </saml:AttributeValue>
       </saml:Attribute>

       </saml:AttributeStatement>
 </saml:Assertion>
```

The SAML attribute statement contains information that uniquely identifies a Shibboleth session between an identity provider and a service provider for one user by means of the session handle that is encapsulated in the <saml:NameIdentifier> element. The attribute name is encapsulated in the <saml:Attribute> element, followed by its value encapsulated in the <saml:AttributeValue> element.

2.10.3 SAML Authorization Decision Statement

The SAML authorization decision statement is built using the attribute statement where the decision takes the name of a specific attribute and its value represents the actual decision.

All of the described SAML assertions need to be protected by the virtual organization's agreed upon policies that take the form of a group of trusted CA (Certificate Authorities) that issue trusted public key certificates for every institution. SSL (Secure Socket Layer) is used by Shibboleth to encrypt and transport the information. A <ds:Signature> element is used to encapsulate the signed SAML statements exchanged by the identity provider and the service provider in the HTTP protocol as shown in the example below:

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:SignedInfo>
      <ds:CanonicalizationMethod
 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      <ds:SignatureMethod
 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <ds:Reference URI="#c7055387-af61-4fce-8b98-e2927324b306">
      <ds:Transforms>

      <ds:Transform
            Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
       <ds:Transform
             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
       <InclusiveNamespaces
             PrefixList="#default saml samlp ds xsd xsi"
             xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"/>
       </ds:Transform>

       </ds:Transforms>

       <ds:DigestMethod
       Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
       <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
       </ds:Reference>
       </ds:SignedInfo>

 <ds:SignatureValue>
```

```
x/GyPbzmFEe85pGD3c1aXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5
EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen3CP+m3D
w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
</ds:SignatureValue>

<ds:KeyInfo>
<ds:X509Data>
<ds:X509Certificate>
MIICyjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakxCzAJBgNVBAYTAlVT
MRIwEAYDVQQIEwlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
F1VuaXZlcnNpdHkgb2YgV2lzY29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
bmZvcm1hdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgQ0Eg
LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsx
CzAJBgNVBAYTAlVTMREwDwYDVQQIEwhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
Ym9yMQ4wDAYDVQQKEwVVQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk
dTEnMCUGCSqGSIb3DQEJARYYcm9vdEBzaGliMS5pbnRlcm5ldDIuZWR1MIGfMA0G
CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTx8vuRay+x50z7GJj
IHRYQgIv6IqaGG04eTcyVMhoekE0b45QgvBIaOAPSZBl13R6+KYiE7x4XAWIrCP+
c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
qgi7lFV6MDkhmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX
8I3bsbmRAUg4UP9hH6ABVq4KQKMknxu1xQxLhpR1ylGPdiowMNTrEG8cCx3w/w==
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</ds:Signature>
```

The <ds:Signature> element of SAML statement is used to contain the value of the signature of the exchanged statements in order to uniquely trace them to an identity provider or a service provider. The usage of certificates and public-private key encryption enforces the agreed upon policies that make the structure of a federation.

2.11 Attribute Management

The wide use of Shibboleth in the academic community has led to the need for a structured way of managing the attributes that each user needs to have in their identity management system or anywhere else their attributes are stored. The EDUCAUSE/Internet2 [INT2] has defined eduPerson - an object class used by campus directories (e.g., LDAP, Windows Active Directory, etc.) in order to facilitate inter-campus communication and resource sharing. More information about the eduPerson

object class can be obtained from [EDUPERSON]. The eduPerson object class defines auxiliary attributes needed by the academic community such as an individual's institutional affiliation or access rights that he or she might possess to access various resources in order to more easily implement inter-campus resource sharing.

The eduPerson object class contains: eduPersonAffiliation, eduPersonNickname, eduPersonOrgDN, eduPersonOrgUnitDN, eduPersonPrimaryAffiliation, eduPersonPrincipalName, eduPersonEntitlement, eduPersonPrimaryOrgUnitDN and eduPersonScopedAffiliation attributes. The eduPerson attributes are intended to support inter-realm applications such as controlled access to web pages or shared resources. Due to the single-value nature of the attributes, and due to the need for fine-grained authorization, the "eduPersonEntitlement" attribute has been used as a holder of fine-grained access control values. The multi values stored in the "eduPersonEntitlement" attribute are currently used to store the user's attributes in the identity provider's identity management directory.

The "eduPersonEntitlement" is the only attribute whose values are not already defined. The attribute "eduPersonPrimaryAffiliation" has only a few values that it can take: faculty, student, staff or member and these values cannot be changed at will. Whereas for the "eduPersonEntitlement" attribute we can have the values "urn:mace:greatplains.net:repository" or "urn:mace:greatplains.net:biosci" that can be redefined whenever necessary according to an established structure.

In order to have a structured expression of the entitlements stored in the eduPerson's eduPersonEntitlement attribute field, the Great Plains Network (GPN) has registered the name "urn:mace:greatplains.net" with the Internet2 Middleware Architecture Committee for Education (MACE). It represents the top level of a hierarchical name space controlled by GPN. The name space "urn:mace:greatplains.net" has been used as the first part of an entitlement's value[MACEGPN]. The second part of the entitlement field is composed of attributes values that represent the desired access control attributes a user must have in order to be authorized to use a resource.

As used by the Great Plains Network, attributes and their values are used in a two level decision making process at the service provider. The first level decision is associated with the web server at the service provider, verifying the existence of an attribute by a user requesting access to the service. The second level of decision making process is made by the actual service provider to authorize a user request based on a value of one or more attributes.

For instance, in the entitlement value "urn:mace:greatplains.net:*repository*" the first part - represented underlined - is used by the web-server to check this value as belonging to the GPN virtual organization. The second part - depicted in italics - is checked by the service provider in the access control decision making process. The first part (urn:mace:greatplains.net) of an entitlement is used in the Shibboleth environment to authenticate the users by the service provider. If the name space's value is not present, the service provider's web server denies access based only on that piece of information.

The second part of the attribute value is used to determine the users belonging to a specific group that is authorized to access a particular resource. Details of this process are described in a paper by Apon, et al [SCPE].

This mechanism creates a viable solution to the problem of authorizing users from various institutions and with different affiliation statuses to access a resource that one particular institution offers. As such the virtual organization (see Glossary) can allow access to a selected number of faculty, students and staff from various institutions to access services without granting access to everyone from those institutions. Using this object class (eduPerson), fine-grained authorization can be used to successfully attain its full potential.

2.12 Identity Provider vs Service Provider Entitlements Management

The Shibboleth architecture requires the identity provider to store the virtual organization's entitlements in its own secured identity management system. This approach has a major drawback in the Shibboleth's requirement that the user's entitlement information for a virtual organization be stored in an institution's identity management directory. Some institutions are always reluctant to allowing outside access to their identity management directory due to user privacy policy restriction. On the same level, the identity provider is not interested in the computational and network overhead incurred by the communication with a service provider in order to exchange attributes. Most importantly, the identity provider does not want to allow users from outside the institution to have access to its secured identity management records and modify them at

will. This issue is extremely problematic for users that belong to particular institutions, but at the same time they may take part in a variety of virtual organizations that their home institution is not member.

On the other hand, the service provider wants to have full access to its entitlements and to have the ability to define, decide and control who accesses their shared resources. Also the communication overhead that is incurred with the identity provider can be eliminated.

In order to satisfy both parties' requests we need to have them work together in the realm of the virtual organization by sharing responsibility commensurate to their role in the federation. The identity provider should be charged only with authenticating users and communicating to the service provider when such an event takes place. The service provider should be in charge of controlling, defining and deciding which user gets what entitlements to access its services. As such the authentication and authorization decisions need to be reached in a distributed fashion, with the identity provider in charge of authentication and the service provider in charge of authorization.

2.13 Shibboleth Advantages and Shortcomings

Shibboleth's purpose is to offer a technology that allows for inter-institutional authentication and authorization services. Of utmost importance for Shibboleth's design is the concept of trust between the participating institutions. Shibboleth offers an infrastructure that builds and supports a web of trust among the institutions of a virtual

organization. Combined with the right policies, Shibboleth offers a seamless medium for institutions to share resources using one localized set of credentials.

One of the most important advantages of using Shibboleth derives from its simplicity in deployment and its concern with total user and service security. Devised with the goal of standards compliancy has allowed Shibboleth to be easily adopted by the community at large. Through the use of attributes Shibboleth offers a fine-grained method of devising and reinforcing an access control policy for each resource that is offered to the virtual organization.

The eduPerson object class that defines an academic persona and is designed to be used in inter-institutional environments offers a lot of attributes that can only hold single values which at times seem hard to implement. The only attribute that was meant to hold multiple values not previously defined is the “eduPersonEntitlement” field. This attribute has been used to store a string of concatenated entitlement values. Due to the large number of entitlement values and the diverse group of institutions that offer a shared computing resource, the storage of diverse entitlement values for a particular user needs to be performed by the service provider’s administrators in order to avoid the stress on the identity provider’s system administrators and limit the possibility of non-authorized access to the resources. The Shibboleth limitation deals with the various administrative and best practices business and privacy policies that govern the member institutions of a virtual organization. Subsequently, the vast majority of these institutions are reluctant to allow people from theirs and other institutions to make changes in the identity

management directory on behalf of the various service providers, besides the authorized personnel.

Shibboleth, as defined, may be perfectly satisfactory when an access decision is based solely on the user being authenticated and possess one or more of the predefined attributes already existent in the eduPerson object class. However, fine-grained access control decisions solely based on attributes belonging to an independent virtual organization or federation is very difficult and unmanageable.

Other approaches have been devised by Internet 2 Middleware Initiative in order to deal with the fine-grained authorization problem, more specifically Grouper (http://middleware.internet2.edu/dir/groups/grouper/) and Signet (http://middleware.internet2.edu/signet/). Grouper enables both automated and manual mechanisms for assigning and managing assignments of users to groups based on their individual campus affiliations, status, or other relevant roles. Signet provides institutions an easy to use framework to manage user access privileges (entitlements) and provides a consolidated, shared authorization data repository that is independent of any specific institutional system. Grouper and Signet can be used separately or together. When used together they enable a distributed model for control, so that the people in charge with assigning or delegating users' access privileges can directly manage them across the campus. Together, Signet and Grouper implement a centralized repository of access privilege entitlements that allow users from various groups or departments to be granted and denied rights to various resources across the campus.

The main problem with the two technologies relies in their dependency on the eduPerson object class to define and store their users' attributes. The eduPerson object class does not allow for virtual organization-related entitlements to be defined in its structure. Signet and Grouper can be used in inter-institutional authorizations as long as the required attributes are part of the eduPerson class. As such, our goal of providing an entitlements repository that would serve an entire virtual organization - no matter what institutions it encompasses - cannot be implemented using these technologies. A new approach is presented in this thesis that creates an autonomous repository of entitlements that belong to any virtual organization and cannot be stored in an eduPerson object. The centralized approach to virtual organization-related entitlements allows for fine-grained access control capabilities to all interested service providers. We hope that the presented prototype or its ideas will be – in the long term – implemented in the Signet and Grouper technologies and become part of the Internet2 middleware software.

The new approach allows for finer-grained authorization to a resource avoiding at the same time the privacy issues associated with public-wide access to an institution's identity management directory. The solution proposed in this thesis allows for a database-based service to be run somewhere in the virtual organization environment and all participating institutions can connect to the server and retrieve information as needed. The entitlements server – proposed in this thesis - allows for entitlements to be stored in a secure fashion on any secure machine accessible from any institution member's service provider of any virtual organization. The SHAR (Shibboleth Attribute Requester)

redirects its attributes requests to the entitlements server's client application that runs in every service provider instead of the identity provider's attributes authority. This way, the entitlements pertaining to a virtual organization are stored in a centralized location accessible from anywhere and each institution that wants access for its members has a designated administrator that can add or delete entitlements for their own users in the service's database and at the same time query user entitlements. The SHAR (Shibboleth Attribute Requester) asserts entitlements to our client application that connect to the entitlements server and under the control of a time window, sends queries to the server which responds with a yes or no answer not allowing for more information to be revealed about the user.

The revised protocol to create the autonomous entitlement service is shown in Figure 3.

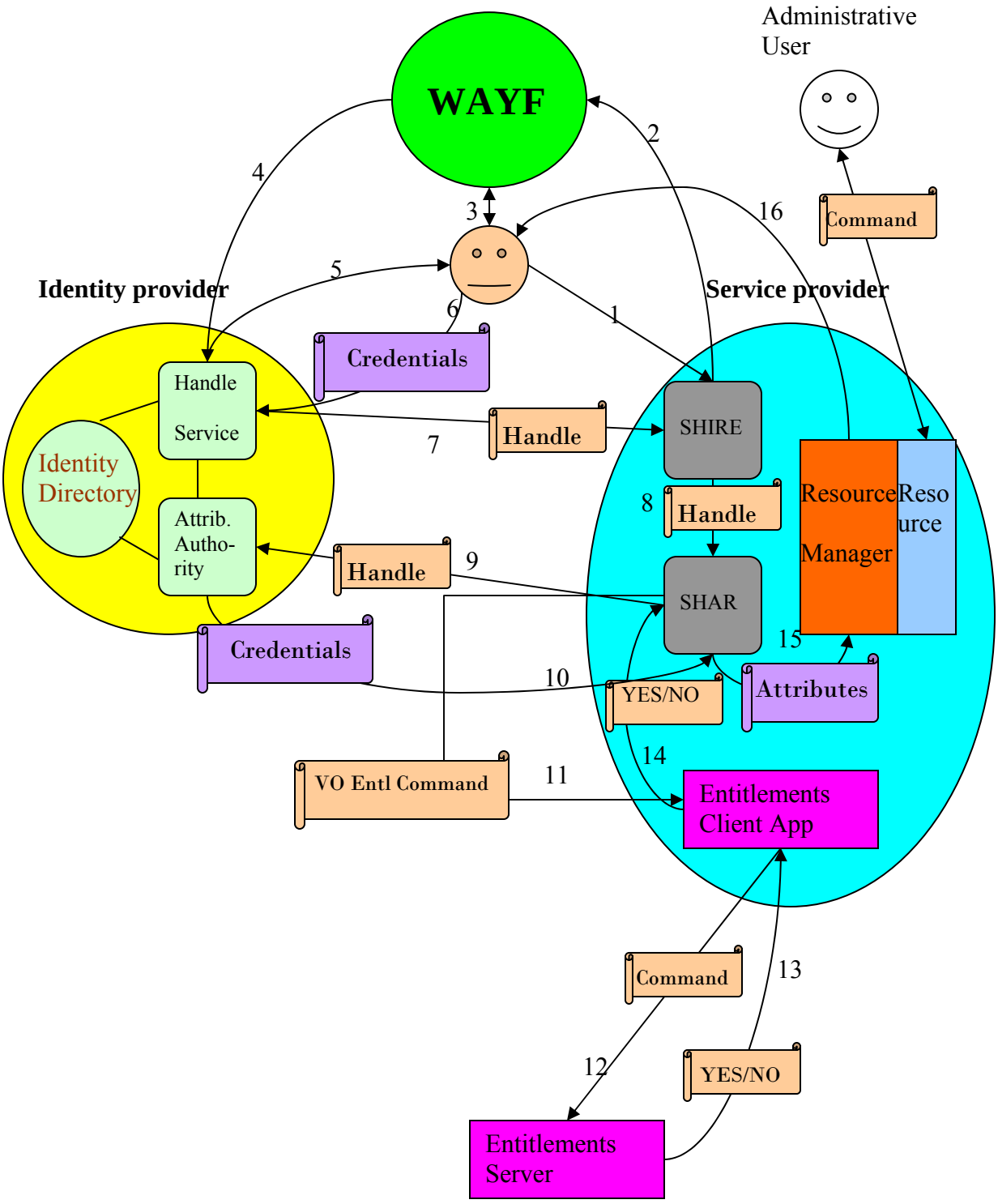

Figure 3. Shibboleth and Entitlements Server protocol outline

The scheme is exactly corresponding to the Shibboleth scheme until Step 10 where besides inquiring the identity provider's attribute authority about the entitlements that a user holds, the queries are also sent to the entitlements server via the entitlements server's client application in Step 11 and the yes/no answers come back in Step 13. This way the service provider can take full advantage of all the attributes that are stored by the identity provider's identity management system and also take advantage of the virtual organization wide entitlements that are stored in the entitlements server. By combining the two sources of information, the service provider can make a fine-grained authorization decision.

There are only two kinds of entities that can request service from the entitlements server: the service providers that run the entitlements server's client application and users who have administrative privileges with the entitlements server through a web-based interface that uses the service provider's entitlements server client application. In order to start communicating with the entitlements server, the service provider needs to establish a trusted communication channel with the server.

In step 11, in order to issue commands, the service provider needs to be authorized by the entitlements server to use its services. Upon completion of the authorization phase, the trusted channel is established between the two parties during a setup phase that stores a symmetric key on the service provider side of the communication as well as on the autonomous entitlements server side. Besides the encryption key, the entitlements server stores the time stamp of the creation of the secure channel. If the service provider does

not communicate again before a predefined amount of time elapses, the next time the service provider communicates with the entitlements server, the server sends a "SESSION EXPIRED" message and does not allow any more communication on the previously established secure channel. Upon completion of a successful setup phase, the service provider has the possibility of sending queries to the entitlements server.

Steps 12 and 13 are used by the client application to send queries and receive yes/no answers from the entitlements server. The service provider can send queries to the server in order to establish if a specific user has all the required entitlements to access a shared resource. Every time the service provider contacts the entitlements server, the server resets the service provider's time stamp. This way, the service provider is always able to access the entitlements server's services for a specified amount of time.

Besides the service provider, some authorized users are also able to access the entitlements server in order to administer its entitlements database. The communication between the authorized users and the entitlements server takes place over the secure channel established by the service provider. As such, before gaining access to the web interface that allows contacting the entitlements server, the users need to be authenticated by Shibboleth, and once their identity has been established, they can try to access the autonomous entitlements server.

The service provider communicates with the entitlements server and sends queries on behalf of the user - if the user has the right authorization to access the entitlements

server's services. If the user becomes authorized to use the server, the server creates a new entry in the time stamps database in order to allow the user to use its privilege for a specified amount of time. The session time is reset every time the user uses the server's services. If the authorized user misses the time window, the session expires and they need to be authorized again through the service provider.

The authorized users are able to add, delete, lookup or display entitlements based on their institutional and virtual organization affiliation. Theses operations are restricted to the institution and virtual organization that the user belongs to. For instance, a user that comes from University of Missouri that is a member of the Great Plains Network virtual organization is not allowed to delete or add user's entitlements that denote a different institutional and virtual organizational affiliation.

3. System Design

3.1 System Design Guidelines

The fine-grained authorization scheme presented in this thesis is designed in accordance with the principles of user data complete security, fast and efficient response time and ease of use. Complete user data security is attained through minimal exposure of the user's data to potential attackers and cryptanalysis through a communication protocol that emphasizes the use of industry-strong standards of encryption and minimizes the actual transfer of data across the communication channel. The efficiency in response time is achieved through a carefully designed structure of the records database and through the use of a readily-available server access policy. The ease of use of our system is attained through careful user interface design that allows the performance of the necessary functions without revealing anything that the current level of privilege of a user does not allow the user to access.

3.2 Data Security

The security of the user data stored in the service's database is of high importance in the design of the entire authorization system. In order to achieve a high level of security, the user data needs to be stored in a secure environment and exposed as little as possible. The transfer of actual user data from the server to the user is minimal and the security of the communication protocol and the communication channel is achieved through a careful selection of technologies that protect the data and still allow for a seamless experience. In

order to achieve the goals of data and communication channel security a few steps have been taken in order to ensure a powerful protection scheme.

The initial step in accessing the entitlements server is to authenticate and authorize the service provider that initiates the communication. The authentication of the users with administrative privileges that want to reach the entitlements server via the service provider is done by the Shibboleth identity provider (IdP) and upon successful completion of the authentication phase, the administrative users need to be authorized by the entitlements server via the service provider. In order to implement the secure communication between the service provider and the entitlements server, the design choice has been to provide an authorization scheme based on the Public Key Infrastructure System (PKI). The entitlements server's administrators generate a public-private key pair for each participating service provider. The server retains for itself a record of the service provider's public key and gives the private key to the corresponding service provider. This system allows for the entitlements server's administrators to have a complete view of the servers' service providers.

The same authentication and authorization scheme could have been built using a user-password model in which the entitlements server keeps a record of the service provider's username and their corresponding password, but the same amount of communication takes place between the administrators of the entitlements server and the service providers. RSA has been chosen due to its encryption strength, its security record and because the use of a username/password scheme requires the establishment of a secure

channel before the actual password is sent by the service provider to the server by the means of a Diffie-Hellman scheme. We found this method to be less straight forward and prone to security breaches due to the problems associated with passwords such as passwords being stolen because of user carelessness while typing.

A perceivable shortcoming of the authentication and authorization scheme - based on public-private keys being distributed to users - arises from the overhead to be generated by creating and distributing keys to various service providers across the virtual organization. This design choice has been made due to the relative small numbers of the entitlements server's service providers. Our entitlements server is used by the administrative users of the various virtual organizations that have users using shared resources and also by the service providers as a step in the Shibboleth authorization process. Due to the high level of security that needs to be available in order to protect the user data, this solution was chosen over the use of usernames and passwords because of the increased security added by encrypting a message twofold. As such, only the service providers that are using the entitlements server need to setup a keys distribution scheme in order to get the necessary keys to communicate. This approach also achieves the benefits of mutual authentication, non-repudiation and data integrity.

The system has been designed using an asymmetric encryption scheme using a 1024 bits long key for the public-private key pair belonging to the service and a 2048 bits long key for the public-private key pair belonging to the service providers. Both key lengths are providing a high level of cryptographic strength. Due to the size of the keys, a brute force

attack needs too much time to complete, rendering the information useless. The shortcoming of using the RSA [RSA] technology resides in its inherent slowness due to the large number of mathematical computations that need to be performed in order to encrypt or decrypt a message. For the purpose of this application, the advantage of being able to uniquely identify the sender and the receiver of a message is much more important than the speed disadvantage.

Eliminating the chance of using frequency cryptanalysis, known-plaintext attacks or other types of cryptanalysis attacks based on the encryption of the same message repeatedly has been a major concern for the design of this system. To alleviate this problem we have added a randomly-generated piece of data to the messages that gets repeated a lot such as the yes/no answers coming from the entitlements server. The random message is generated – as pictured in the Figure 4 below - from a seed and a variable length string dependent on the service provider's username, combined with the current time and the running process's id. A random value is generated using the provided seed value and it is used together with the other values to create a hash digest. The hash function is applied a number of times in order to thoroughly mix the bits and get a value as random as possible.

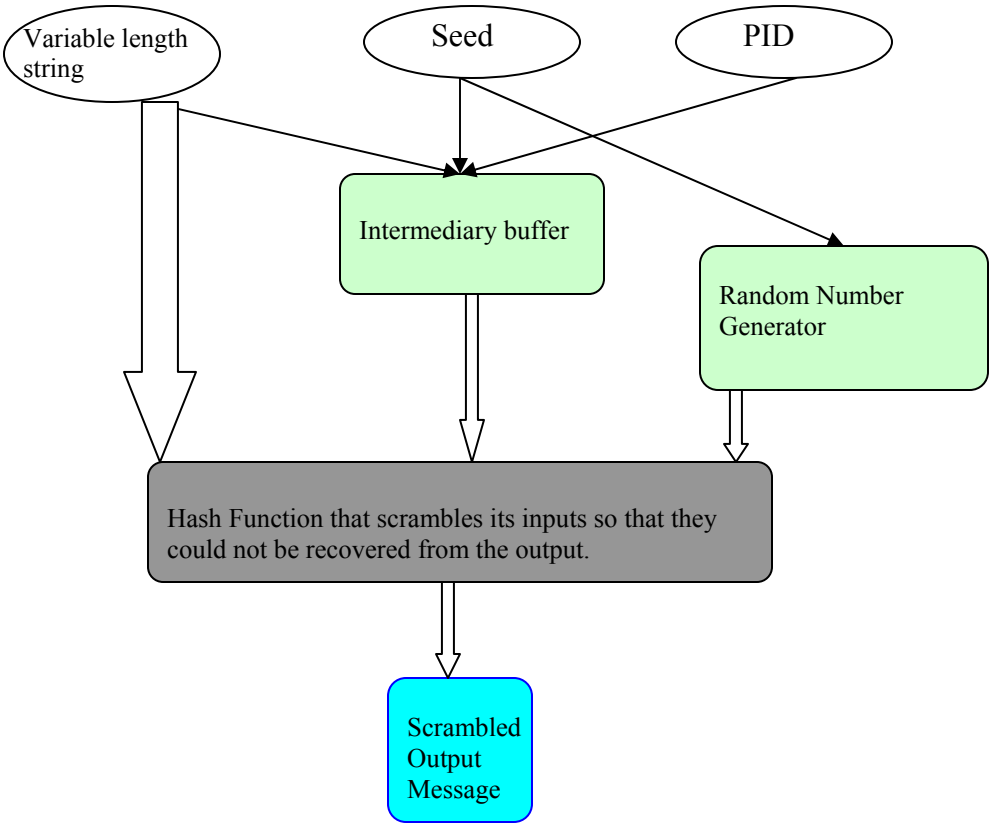

Figure 4. Depiction of the scrambling routine

The use of the 3DES symmetric encryption algorithm in Cipher Chaining Block (CCB) mode also brought a new level of security due to the internal encryption of 64 bits long segments of data using the previously encrypted data or an Initial Vector (IV) to start the process. The CCB mode is one of the modes of operation of symmetric encryption ciphers as depicted in Figure 5 below. In the CBC mode, each block of plaintext is XOR - ed with the previous ciphertext block before being encrypted so that each ciphertext block is dependent on all plaintext blocks processed up to that point.

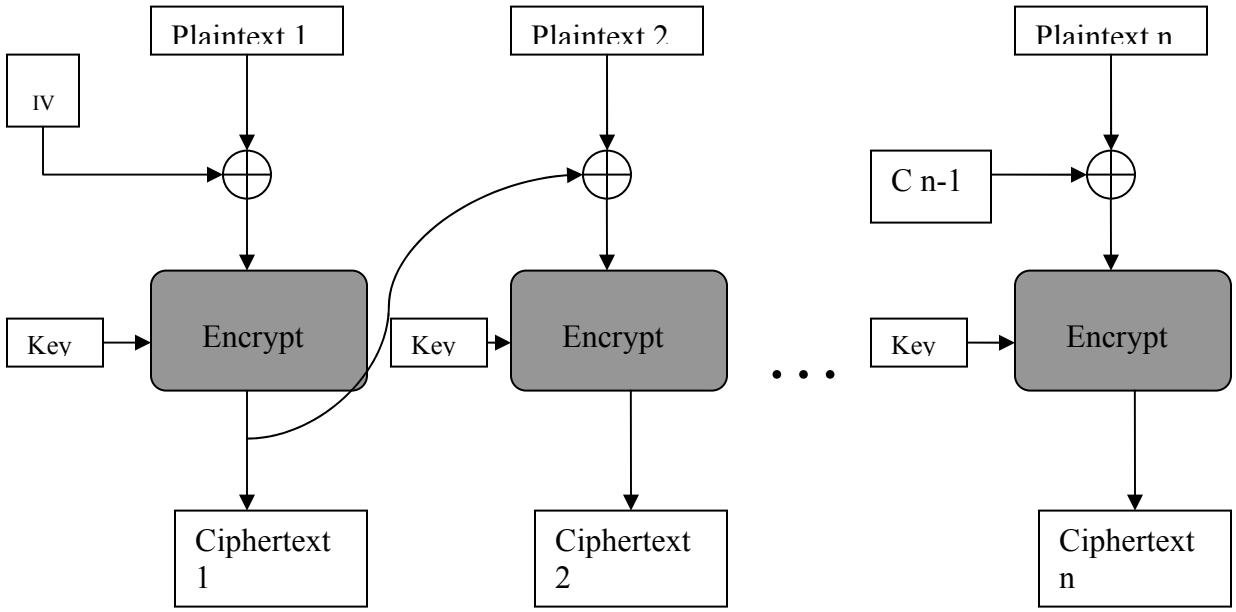

Figure 5. Cipher Chaining Block depiction

The communication channel between the server and the service provider is secured using a symmetric encryption technology – 3DES [3DES]. Once the authentication and authorization phase between the entitlements server and the service provider is successfully satisfied, the actual communication takes place over a symmetrically encrypted data channel. The choice of a symmetric scheme is employed because of the speed advantage over the asymmetric encryption schemes and because of the high level of cryptographic strength exhibited by these algorithms. The choice of 3DES is due to its longevity and integrity.

3.3 System Access Privilege Levels

The users who have access to our entitlement server are differentiated in the following three categories:

- USER access: Service providers need to assert a potential user's entitlements in order to completely determine if a user can be granted access to its computing resources or not. The service providers have access to the lowest priority privilege level and as such they just assert a user's name, home institution, virtual organization and entitlement and receive a yes/no type of answer. Based on the answers obtained from the entitlements server, the service providers is empowered to make fine-grained authorization decisions.

- ADMIN access: Administrative users have access to the records of their own institution or virtual organization. They are able to add a new record, delete an already existing record or lookup a specific record of interest to them. The institutional or virtual organization level administrators do not have access to other institutions or virtual organizations' records in order to modify them – they can only search them based on previously known user name, home institution, virtual organization and entitlement information. The administrative users have access to a few lookup functions that allow them to see all the user and entitlements from their own institution and virtual organization.

- ROOT access: The root level administrative users have access to the entire database and are able to add or delete a new record and also search the entire database for entitlements data. The root level access privilege is only available to a few specific individuals who need to administer the entire entitlements server. They can come from any member institution of any of the service's virtual organizations.

3.4 System Communication Protocol

3.4.1 Protocol Overview

The communication protocol is designed using the client-server methodology and as such there is a server application waiting for a client application to connect and request services.

The communication is always started by the entitlements server's client application. Initially, the client application establishes a secure communication channel between its service provider entity and the entitlements server. The service provider starts the client application by providing its authentication credentials and specifying the type of operation requested – setup of a secure channel. This first step is described in Figure 6 in Step 1. Following the acceptance of the service provider's credentials, the client application builds an authentication message whose purpose is to authenticate the service provider to the entitlements server. Using the previously agreed upon asymmetric keys, the authentication message reaches the entitlement server in Step 2.

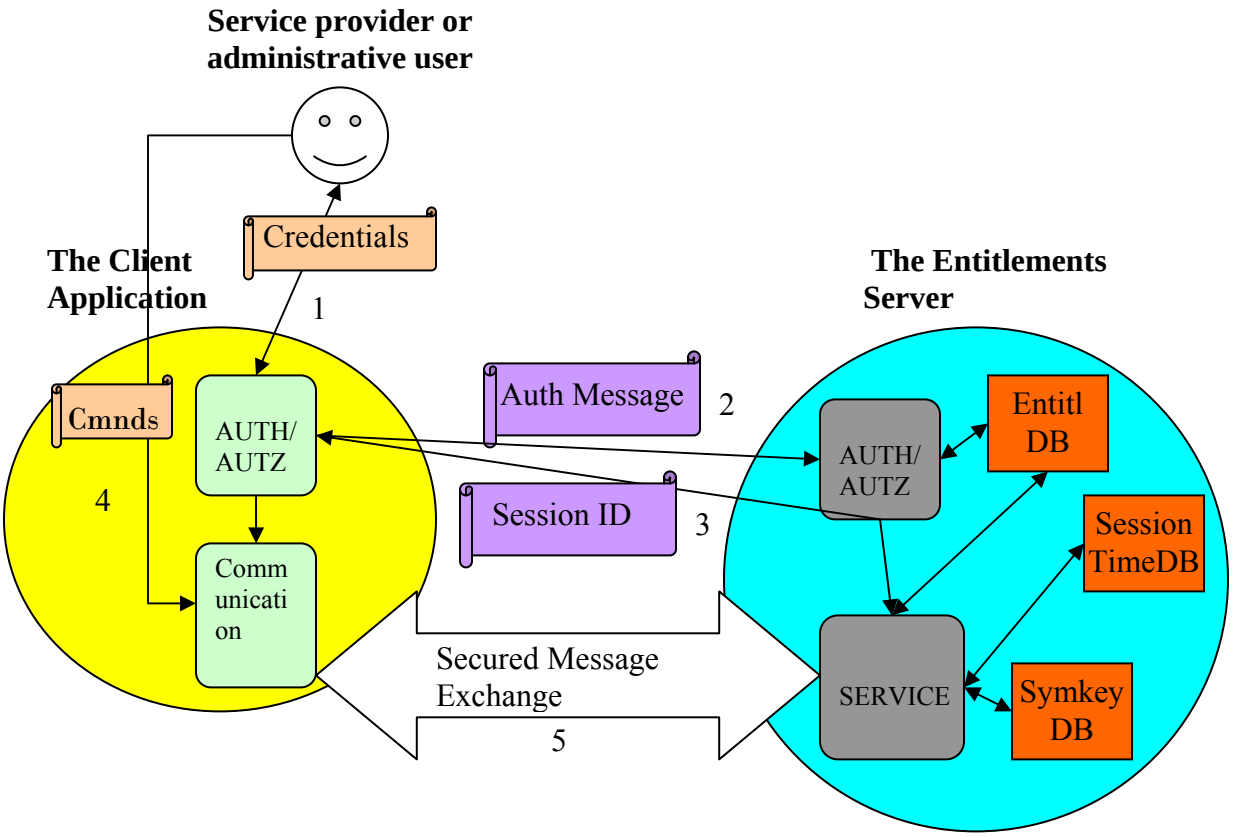

Figure 6. Entitlements Server communication protocol

The entitlement server receives the authentication message and decrypts it, retrieving the service provider's credentials information. The server interrogates its database and based on the result of the query, it decides if the service provider is authenticated and authorized to use its services. The response is sent back to the client. The authentication and authorization steps are completed.

In Step 3, the entitlements server creates a session ID to uniquely identify the newly created session and sends it back to the client application to be used in all subsequent communication. Also it saves the session ID in the SymkeyDB database in order to be

able to retrieve it for use in the ensuing communication. The time stamp of the creation of the session ID is saved in the TimeDB database in order to accurately keep track of the time windows that the service provider is allowed to use the newly created session ID to communicate with the autonomous entitlements server.

Upon completion of the authentication and authorization phase, the service provider is able to send queries to the entitlement server in Step 5. The server returns yes/no answers to each of the queries.

When an administrative user arrives at the service provider after being authenticated by Shibboleth, the service provider uses the already established secure communication channel with the entitlements server to authorize the user to the entitlements server. The server decides if the user is authorized to use its services and in case of an affirmative resolution, it creates an entry in the TimeDB database to record the time of the authorization decision. The administrative user that has been authorized by the entitlements server has only a finite window of time to use the server. This time window is reset every time the user sends commands to the server.

3.4.2 Protocol Steps Description

Step 1. Session begins:

The service provider launches the client-side application of the entitlement server. The client application requires the service provider to provide the name of the entitlements

server it tries to reach, followed by the service provider's username, home institution, the institution's virtual organization and the type of operation it requests. SP_SETUP is the operation used initially to set up a secure communication channel between the service provider and the entitlements server. SP_LOOKUP is the operation used by the service provider to query the entitlements server. SP_USE is the operation used to piggy back administrative user's updates and queries to the entitlements server.

Step 2. Service provider authentication and authorization:

Once all user data has been gathered, the client application verifies the correctness of the service provider's input and formats it into a predefined authentication message format. The authentication message is scrambled in order to allow for a first level of security.

The client application sends the length of the authentication message along with a descriptor that uniquely identifies the service provider to the entitlements server. This unique identifier allows the server to locate in its directory structure the public key of the service provider that pretends to have the corresponding private key. The authentication message is RSA encrypted using the server's public key - stored in the client's directory.

The sent message has the following format:

Message_to_be_sent = $E_{\text{Server public key}}$(message)

This one way authentication method assures that the RSA encrypted data is readable only by the entitlements server as it owns the private key that pairs with the public key used to RSA encrypt the message. As only the entitlements server has access to the private key, the user can be assured that the information sent over the communication medium can only be RSA decrypted by the particular server. The server RSA decrypts the message with its private key and makes sure it can accommodate the service provider's authentication message. Also the server locates the service provider's public key in order to be able to RSA encrypt the session ID it creates in the case of a positive authentication and authorization decision.

The client application encrypts the scrambled authentication message using the RSA public-private key encryption scheme. First, the authentication message is RSA encrypted using the entitlements server's public key. Second, the previously resulted ciphertext is RSA encrypted one more time using the service provider's private key and the authentication message is now sent to the entitlements server.

The entitlements server receives the agreed upon number of authentication messages from the service provider and decrypts them sequentially building a complete authentication message. For instance, if the authentication message is 128 bytes in length, the entitlements server is notified that it needs to allocate two times 64 bytes in order to properly store the authentication message. After the server allocates the necessary memory, it receives 2 messages containing 64 bytes that represent the authentication message.

First, the received ciphertext is RSA decrypted using the service provider's public key. Second, the resulted ciphertext is RSA decrypted once more time using the entitlements server's private key. The application was designed with the highest level of data security in mind and as such the two-level RSA encryption/decryption scheme used allows for both parties (the user and the server) to authenticate each other.

At this moment both the service provider and the entitlements server are assured that the authentication message was generated by the correct party. The format of the authentication message is described below:

$$\textbf{Message\_to\_be\_sent} = E_{\text{User's private key}}(E_{\text{Server's public key}}(\textbf{authentication message}))$$

The entitlements server descrambles the authentication message and queries its entitlements database. In order for the user to be granted any type of access, the particular user needs to have a previously available record in the server's database that gives the user such a privilege. Once the authentication decision has been reached by the entitlements server, it creates and sends back the yes/no type of decision.

Step 3. Session ID creation

If the authentication decision was yes, the server generates a unique session ID for this current transaction with the service provider. The unique session ID is comprised of an 3DES IV (Initial Vector) and a symmetrical key. The session ID is used subsequently to communicate with the service provider. The communication is encrypted using the 3DES symmetrical encryption technology. Once the session ID is generated, the server double RSA encrypts it using its own private key and the service provider's public key. The resulting ciphertext is sent over the communication medium to the client. From this point on the communication channel between the client and the server is symmetrically encrypted taking advantage of the speed and performance of this cryptographic technology.

The session ID is also stored in the entitlements server's SymkeyDB database so that the server can retrieve it when a new communication with the same service provider is established. The entitlements server also saves the time stamp of the session ID's creation in the TimeDB database. It uses the time stamp to decide if the service provider should still be allowed access to its service beyond an established amount of time.

Step 4. Communication based on the authentication results and the session ID

The communication between the service provider and the entitlements server takes place over the secure communication channel established in the previous step.

The service provider can use the secure channel to send queries about users and their entitlements and the entitlements server answers back with yes/no answer. The service provider also queries the entitlements server for authorization decisions concerning any administrative users that want to gain access to the server. Using the already established channel, the service provider constructs an SP_LOOKUP message and sends it to the server. The server retrieves the service provider's symmetric key and checks the TimeDB to determine if the service provider has a valid session. In case of an affirmative answer, the administrative user's credentials are received by the server and an admission decision is reached. The server sends back the decision and creates an entry in the TimeDB database for the newly authorized administrative user with the corresponding time stamp.

Step 5. Messages containing commands are exchanged

The secure channel created by the service provider is used to send SP_LOOKUP queries or to send SP_USE commands.

The SP_LOOKUP commands are used by the service provider to send queries to the entitlement server. These queries can relate to some user's requests to access a shared resource or administrative users who want to have updating roles within the entitlements server.

The SP_USE commands are used once an administrative user has been authorized by the entitlements server to use its services and a corresponding time stamp has been saved in the TimeDB database. The administrative user is able to send update or query commands to the entitlements server according to its privilege level (administrator or root). The entitlements server receives the commands, verifies their authorization and executes them.

3.5 Entitlement Server Commands

The entitlement server can be used by issuing one of the three commands that are discussed in the following three sections.

3.5.1 SP_SETUP

SP_SETUP is the command or operation type used initially to set up a secure communication channel between the service provider and the entitlements server. The process is outlined in Figure 7.

In order to start communicating with the entitlements server any service provider needs to authenticate themselves and get authorization to use the entitlements server from the

server itself. Upon completing the authentication and authorization phase, the service providers receive a session ID that is used in subsequent communication with the entitlements server. The session ID is saved by the service provider in a local file and by the entitlement server in the SymkeyDFB database.

Once the secure communication channel is established, all of the following operations, such as SP_LOOKUP or SP_USE take place over the secure channel secured by symmetric encryption.

The entitlement server's client application returns to the caller some status messages according to the results of the communication with the server. If the service provider's authentication and authorization with the server failed, a value of 0 representing SP_AUTHENTICATION_FAILED is returned, otherwise a value of 1 represented by SP_SETUP_SUCCESS is returned.

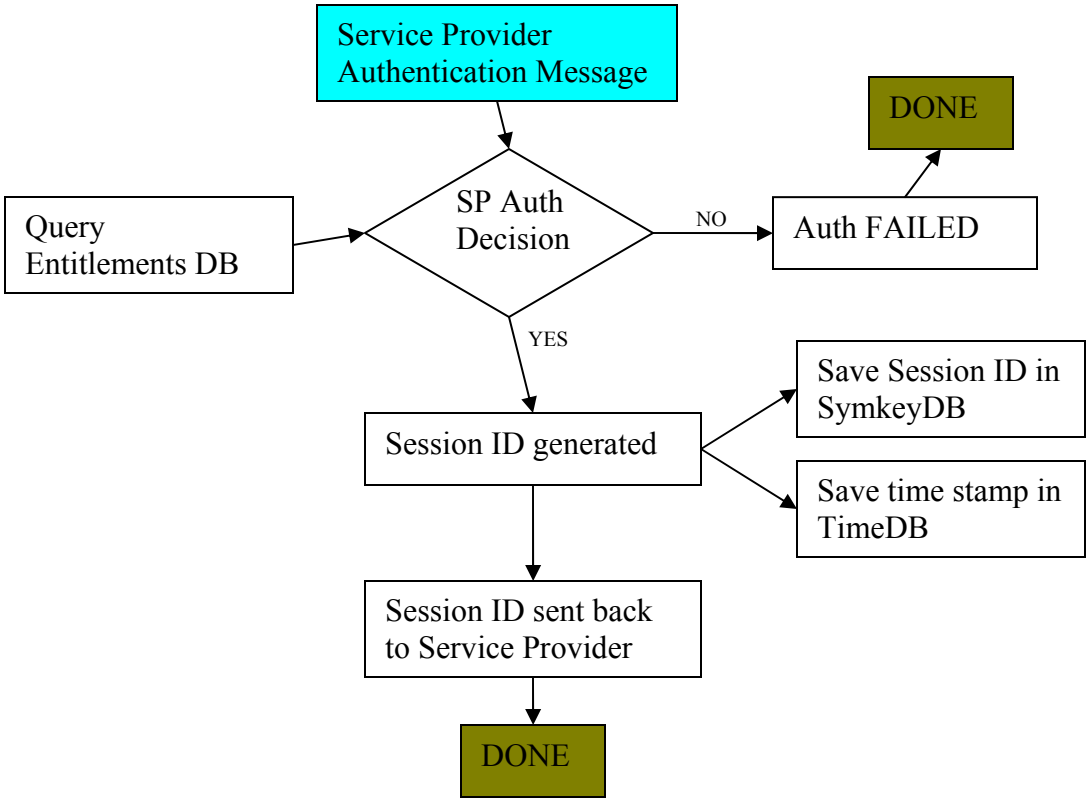

Figure 7. The SP_SETUP operation

3.5.2 SP_LOOKUP

The SP_LOOKUP operation is used by the service provider to send queries to the entitlements server .The process is described in Figure 8. The queries can refer to users who want to be authorized to use one of the computing resources that the service provider is protecting or they can reference an administrative user who wants to be granted access to update and issue commands directly to the entitlements server.

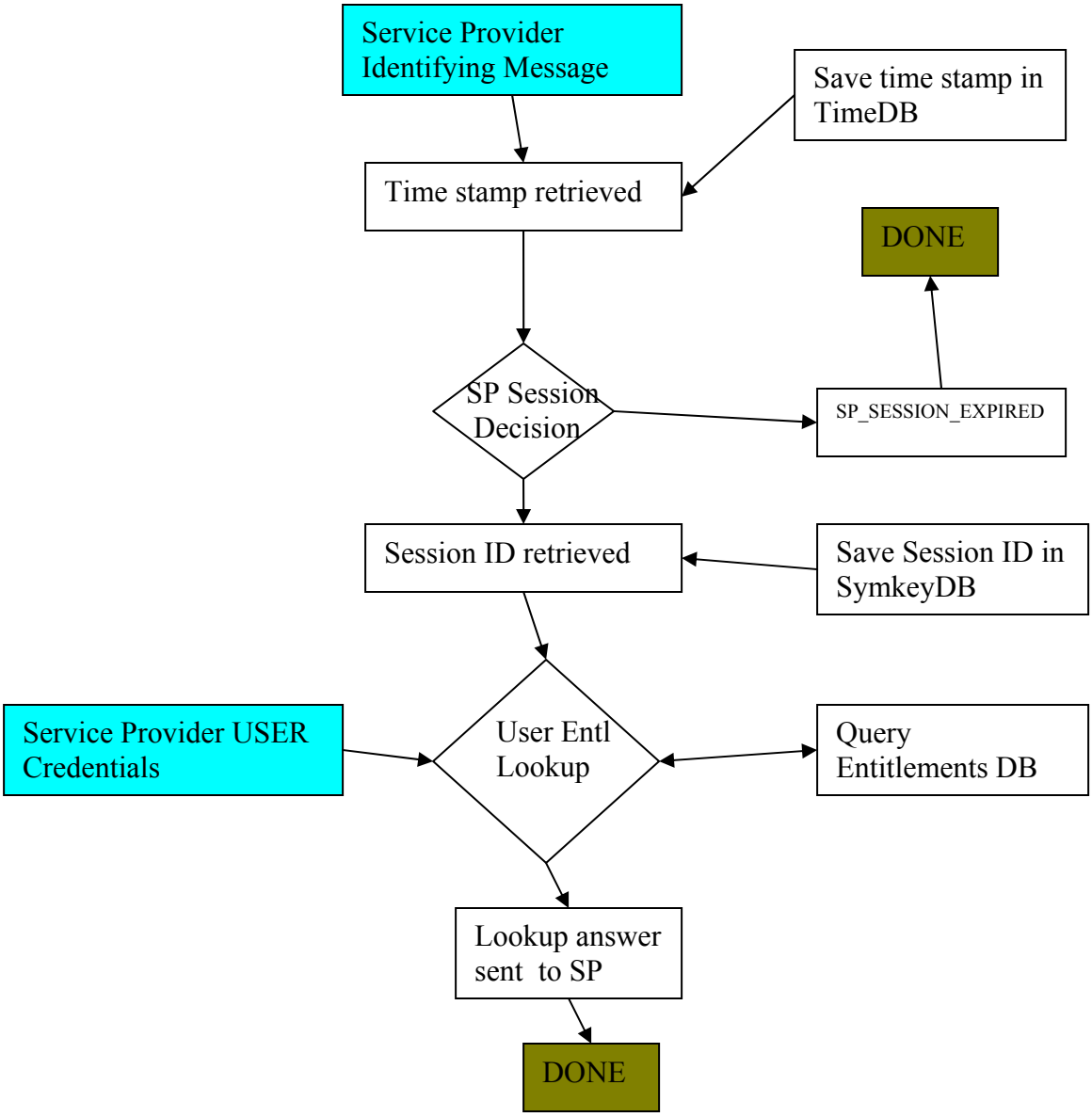

Figure 8. The SP_LOOKUP operation

Upon receipt of the command from the service provider, the entitlements server queries the TimeDB database to find out if the service provider's session has expired or not. In case the session has expired, the server returns an error message, telling the service provider that it needs to establish a new secure communication channel using the SP_SETUP command.

Otherwise, the session ID corresponding to this service provider is retrieved from the SymkeyDB database. The session ID is used to decrypt the credentials of the user to be authorized. The credentials are sent encrypted over the secure channel by the service provider. Upon successful receipt, the server looks up the user in the entitlement database and an authorization response is formed and sent back to the service provider using the same encrypted channel. Also the service provider's time stamp is updated in the TimeDB database to reflect the new time stamp. A new time record is stored in the time database for the user if the user is an administrative user requiring authorization to use the entitlements server.

The entitlement server's client application returns to the caller some status messages according to the results of the communication with the server.
If the service provider's session with the server expired, SP_SESSION_EXPIRED is returned, otherwise SP_SESSION_OK is returned and the TimeDB database is updated with the new time stamp. Following the decision concerning the service provider, the entitlements server decides if the query sent is available in the entitlements database or not. USER_ENTITLEMENT_LOOKUP_SUCCEEDED is sent in case the entitlement and the user are found or USER_ENTITLEMENT_LOOKUP_FAILED in case no such combination is available in the entitlements database. Also, if the user is an administrative user, the positive decision time is logged into the TimeDB database.

3.5.3 SP_USE

SP_USE is the operation used to piggy back administrative user's updates and queries to the entitlements server. The process is outlined in Figure 9.

The service provider starts by sending its coordinates to the entitlement server and waits to receive an answer relative to the status of its session. If the session is not expired, the service provider sends to the entitlement server a message containing the administrative user's credentials. If the administrative user has an unexpired session, the service provider sends to the server the username, institution, virtual organization and operation (add, delete or lookup) to be performed on behalf of the administrative user.

The entitlement server receives the messages and makes sure that the service provider has an unexpired session. If the session is expired, the server deletes the entries pertaining to this service provider from the SymkeyDB and TimeDB databases and sends to the service provider an error message. If the service provider's session is fine, the entitlements server checks the session for the administrative user. If the session is expired, the server deletes the administrative user's entry in the TimeDB database and sends an error message back to the service provider.

If the administrative user has an unexpired session, the server receives and processes the requested command from the administrative user. If the user has the "admin" – administrative - access right, he or she can only modify entitlement records that belong to

their virtual organization and institution. If the user has a “root” access right, the administrative user is able to modify the entire database.

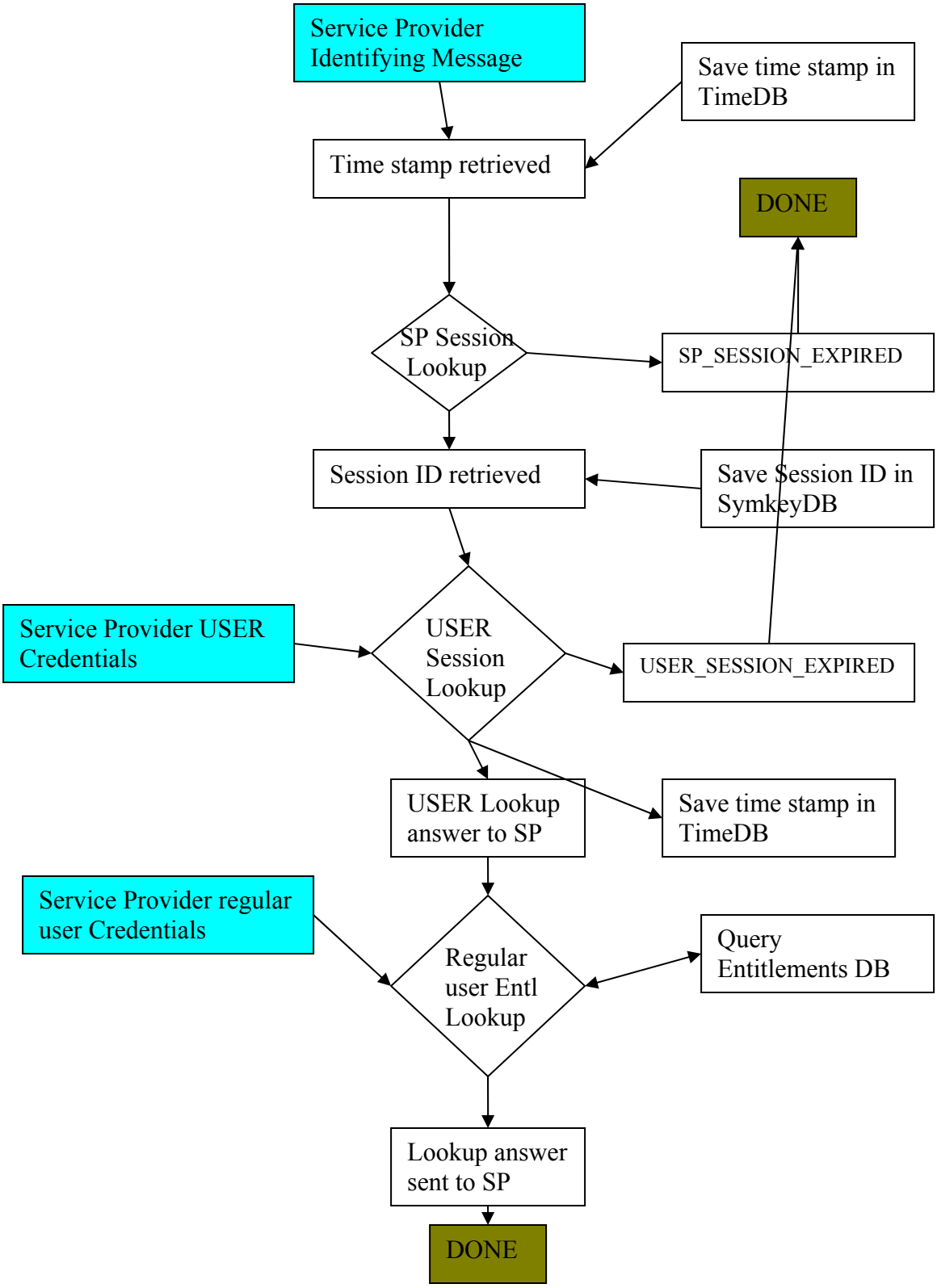

Figure 9. SP_USE operation

Every time a transaction is processed by the entitlements server on behalf of some administrative user, the session time stamp for both the service provider that provides the secure communication and the administrative user are updated in the TimeDB database so that to reset time frame that both have the right to use the already established secure channel. Upon completion of the requested command, the entitlements server sends back a status message describing the success or failure status of the requested operations.

The entitlement server's client application returns to the caller some status messages according to the results of the communication with the server. If the service provider's session with the server expired, SP_SESSION_EXPIRED is returned and the corresponding record in the TimeDB and SymkeyDB are deleted, otherwise SP_SESSION_OK is returned and the TimeDB database is updated with the new time stamp.

Following the decision concerning the service provider's session, the entitlements server decides if the query sent comes from an authorized user. If the entitlement of the administrative user is different from "admin" or "root", the server issues the USER_AUTH_MISSING message and terminates the communication. USER_SESSION_EXPIRED is sent in case the administrative user's session is already expired. In this case his TimeDB record is also deleted in order to record the termination of the session. In case the session is fine USER_SESSION_OK is sent back to the client application. The TimeDB database is updated with the administrative user's new time stamp in order to record the start of a new session.

After an add operation, the server returns one of the three possible error codes: ADD_ENTRY_SUCCESS for a successful addition of a new record, ADD_ENTRY_ALREADY_EXISTS if the add operation failed due to an already existing record or ADD_ENTRY_FAILURE in case of an error due to the database. After a delete operation, DELETE_ENTRY_SUCCESS is returned in case of a successful delete operation or DELETE_ENTRY_FAILURE in case of a failure. Similarly, LOOKUP_CODE_0 is sent back in case the lookup operation failed, otherwise LOOKUP_CODE_1 is sent back to the client application.

4. The Server Implementation

4.1 Server Functionality Outline

The server has been designed to be highly available to potential users, to communicate securely and to provide accurate responses in a timely fashion. The server has been designed modularly having configurable capabilities with respect to the underlying databases. It is comprised of a number of processes that are started by the main process of the server once a new connection from a user is accepted.

The server starts by running an initialization routine that encompasses the actual starting of the application and providing it with arguments. The main process of the server starts by accepting command line arguments passed by the entity that started the server application. These arguments are used to fine tune the server in order to allow for an input file to update the contents of the entitlements database, to set and unset a debug flag that can be used to follow the execution of the server's processes in order to find malfunctions, and to update the port on which the server is listening for new connections.

Once the argument list is parsed, the server starts its execution by creating a network socket to be used by its main process. The main process listens on this socket for client application requests to communicate with the server. The socket is set in the "ACCEPT" state and the server blocks waiting for incoming communication from potential users. Once a request arrives, the server creates a separate process to deal with the client

application request and passes the responsibility on to the newly created process, returning to wait for new requests from potential users.

The new process takes over from its parent and blocks on reading the client's first communication block of data. In order for the communication between the entitlements server and the client to start, the server waits for the client to send an authentication message so that the server can know the identity of the connecting service provider and also the type of operation requested (SP_SETUP, SP_LOOKUP or SP_USE). The communication needs to take place over a secure connection and as such we have chosen RSA to encrypt the client's authentication message using the server's public key available to all client applications running on various service providers.

Once the initial message is received, the entitlements server decrypts it and retrieves the name of the service provider requesting service together with the operation requested. According to the type of operation, the entitlements server takes different courses of action.

4.1.1 SP_SETUP Operation

Firstly, if the operation is a SP_SETUP requesting the entitlements server to authenticate and authorize the newly arrived service provider, the entitlements server searches the entitlements database and decides if the service provider has an entry in its database that corresponds to the service provider's username, institution name, virtual organization and the entitlement value of "user". The "user" entitlement allows for only queries to be sent

to the entitlements server, which is exactly what the service provider needs. If all these requirements are satisfied, the server creates an answer containing a yes/no response. The response to the service provider's authentication and authorization inquiry is RSA encrypted and sent back. The answer is comprised of a yes/no answer and also from a randomly generated string in order to help pad the message before being RSA encrypted. The yes/no answer has a predictable format that allows for easier cryptanalysis of the message; potentially leading to finding the asymmetrical encryption key pair. By inserting a random generated string to these responses, the potential for breaking the cryptographic code is considerably reduced.

By double encrypting (once with the server's private key and once with the client's public key) the response message, both the client and the server can authenticate each other. The server authenticates the client by the mean of the client's public key and the client authenticates the server by means of the server's public key. This way both parties are mutually authenticated and can proceed to exchange information about user entitlements in a secure fashion.

In case of an affirmative response, the server starts the process of creating a secure and faster communication channel with the user. The server creates a unique session ID that is comprised from a symmetric key and an initial vector (IV) to be used in the ensuing symmetric encrypted communication channel. The session ID uniquely identifies the current communication between the user and the server and also serve as the symmetric encryption algorithm 3DES's encryption/decryption key. In order to achieve a fast and

secure communication channel, a symmetric encryption scheme was adopted because of its cryptanalysis resistance strength and speed of encoding/decoding. The session ID is sent to the client using the RSA encrypted secure channel used before and from this point on in the communication, the channel between the user and the server is secured by the use of industrial-strength symmetric cryptography. The key is also stored in the SymkeyDB database on the server side and in a file on the client side as the next time the client is executed, the stored key can be easily retrieved and used to communicate with the server. The time when the service provider becomes authenticated and authorized to use the entitlements server, is saved in the TimeDB database on the server side in order to allow access for a specific and finite window of time. This procedure ensures that the client and the server regularly re-authenticate each other to reduce the possibility of a Man in the Middle (MIIM) attack.

The post-authentication communication between two parties takes place in 1024 byte increments. Each encrypted packet has a 1024 byte limit in order to take full advantage of the underlying IP structure of the networking protocol. A regular Ethernet frame has a maximum 1500 bytes in payload length. The length of our transportation-level packet should be smaller in order to accommodate the TCP header and the IP header and still be under the 1500 byte limit. The design choice of using 1024 byte long packets has been done in order to minimize the number of fragmentations a packet needs to go through before reaching its destination, supposing that the network route the packet is taking is Ethernet-implemented.

4.1.2 SP_LOOKUP Operation

Secondly, if the operation is a SP_LOOKUP message, the server receives the service provider's identity and searches the TimeDB in order to decide if the service provider has been already authorized to use its services. This is proven by the existence of a record in its TimeDB database. If the time stamp is older than the predefined finite window of time, the time record is deleted and the service provider is instructed to come back and issue an SP_SETUP command in order to re-authenticate and be authorized again to use the server. The corresponding SymkeyDB session key record is also deleted.

If the service provider's session is valid, the server retrieves its symmetric key that was established in the previous SP_SETUP operation. The key is used to encrypt the positive session valid response to the service provider. A new time stamp is stored in the TimeDB for the new service provider session that has just started.

Upon receipt of a favorable answer from the entitlements server, the service provider sends the query using the established secure communication channel. The server receives the message, decodes it and runs the query against the entitlements database generating a yes/no response back to the service provider. If the user part of the query is an administrative user, a new record is created in the TimeDB database to accommodate future communication from that authorized administrative user.

4.1.3 SP_USE Operation

Lastly, if the operation is a SP_USE command, the server performs the operation outlined in the SP_LOOKUP operation, but upon receipt of the administrative user's credentials, the server verifies the administrative user's session status. If the user's session has expired, its record in the TimeDB is deleted and a corresponding message is sent to the service provider, terminating the communication. Otherwise, the time record is updated and the server waits for the service provider to send the administrative user's command.

The user command sent from a validated administrative user is executed and a status message is returned according to the nature of the requested operation. These messages include and are not limited to status responses for add or delete a record requests and a yes/no answer for the lookup requests.

The valid types of commands that an administrative user can issue to the entitlements server are: add a new record, delete an already existing record, lookup an user's entitlement, display all the entitlements belonging to one user, display all the users that have a specified entitlement, display all the records in the database or logout the administrative user from the database. These administrative commands are described in the next paragraphs.

4.1.3.1 The ADD Command

The add command is used by the administrative user to add a new record to the entitlements database. A new record is comprised of the user's credentials and the

entitlement value. The user's credentials are described by the user name, the user institution, and the user virtual organization.

Once the command is issued, and the entitlements server has access to the user's credentials and the entitlement value, the server verifies if an entry with the same specifics is already in the entitlements database. In case one already exists, the server sends back to the client an error message that describes the existence of a similar entry (ADD_ENTRY_ALREADY_EXISTS). In case there is no record with the same information, a new record is created and inserted in the database. A message describing the success or failure of the actual record insertion in the database is sent back to the user (ADD_ENTRY_SUCCESS or ADD_ENTRY_FAILURE).

4.1.3.2 The DELETE Command

The delete command is used by the administrative user to remove an already existing record from the entitlements database. A record is comprised of the user's credentials and the entitlement value. The user's credentials are described by the user name, the user institution, and the user virtual organization.

Once the command is issued, and the entitlements server has access to the user's credentials and the entitlement value to be deleted, the server creates a key that describes the record to be deleted. The record is removed from the database. A message describing the success or failure of the actual record deletion in the database is sent back to the user (DELETE_ENTRY_SUCCESS or DELETION_ENTRY_FAILURE).

4.1.3.3 The LOOKUP Command

The lookup command is used by the administrative user to lookup a record from the entitlements database. A record is comprised of the user's credentials and the entitlement value. The user's credentials are described by the user name, the user institution, and the user virtual organization.

Once the command is issued, and the entitlements server has access to the user's credentials and the entitlement value to be searched for, the server creates a key that describes the record to be searched. The search of the key in the database is started. If a record with the requested data is found, a message describing the success of the lookup operation is sent back to the user (LOOKUP_CODE_1). Otherwise a message describing the failure of the lookup operation is sent back (LOOKUP_CODE_0).

4.1.3.4 The DISPLAY Commands

There are three available display commands that are used to create a report against the entitlements database.

The first display command is used to retrieve all the entitlement values of a specified user. The user is specified using its user credentials. The user's credentials are described by the user name, the user institution, and the user virtual organization. If the user has entitlement entries in the entitlements database, the server encrypts the respective entries and sends them back to the client to be displayed.

The second display command is used to retrieve all the users that have a specified entitlement value. The entitlements database is queried and if there are records containing the entitlement value, they are encrypted and sent back to the client application to display.

The last display command is used to retrieve the contents of the database. If the administrative user has “admin” privileges, then he or she is only able to see all the records pertaining to their institution and virtual organization. If the administrative user has “root” privileges, he or she is able to see all the entitlements database’s records.

4.1.3.5 The LOGOUT Command

This command is used by the administrative user to logout from the entitlements server once the administrative tasks have been performed. Upon issuing this command, the administrative user’s record in the TimeDB is deleted leading the server to send USER_SESSION_EXPIRED messages on future attempts to use the entitlements server without prior re-authorization.

4.2 The Server Networking Component

This server listens on a default port 9544. This is the port that all the client applications are expected to connect to in order to communicate with the server. The port number has been chosen in order to comply with the Internet Assigned Numbers Authority (IANA) recommendations. If the server’s administrators want, they can start the server on a different port using the –p starting argument. If this is done, all the client applications

need to be notified of the change in order to update their “key_file.txt” setup files. The “key_file.txt” file is used in the initializing the client application by providing the name s of the public and private key files that that client needs to use (e.g., the client application public key, the client application private key, the server public key) and the server port number. The port number is used to connect to the entitlements server.

The server is configured to accept maximum a number of MAX_CLIENTS clients applications simultaneously. The constant MAX_CLIENTS can be changed as needed in order to accept greater or fewer connections according to the capabilities of the supporting computing platform. The server is running in an endless loop that accepts connections from client applications and starts a new process for every new client connection that has been accepted. Once the new process is started, the server returns to its loop waiting for a new client application to start the three-way handshake process in order to establish a TCP connection. In order to avoid the cases where a software signal interrupts the server process from its wait on the “accept” system call, upon starting, the server verifies if it had been awakened by a software signal and in that case it goes back to sleep by calling the “accept” function call. This approach is needed if the server application runs on a computing platform that doesn’t automatically send the server process back to sleep upon receipt of an interruption signal. For instance, the server is run using a multi-process design and when a process is terminated, a SIG_CHLD signal is raised in order for the parent process to wait for and collect the resources used by the terminated child process. If this signal awakens the main process from its wait on the “accept” system call, we need to make sure that it goes back into the waiting state.

4.3 The Server Multi-processes Design

In order to better serve a large number of client applications, the server runs in a main process and it creates a new process for each new client application that connects to the server. This design choice allows for a larger number of client applications to receive service from the server at the same time without significant delay beyond the capabilities of the underlying networking infrastructure. The choice of using processes over threads has to do with the UNIX implementation of threads as a user-space library in contrast with the kernel-level implementation of processes. The use of a pool of processes for the server to choose from whenever a client connects could provide an alternative way of improving the server's efficiency.

The main process of the server uses parallel executing instruction streams in order to serve as many requests as possible. The parent process creates a new socket for each child process to use in its ensuing communication with the client application. Each child process has a secure and private communication channel while the parent process can continue to listen for new requests without interfering with the request processing associated with each individual child process.

In order to assure graceful starting and stopping of the server's processes, interrupt handlers have been setup in order to avoid abnormal and unexpected termination of the server and also to allow the server to collect the resources used by all of its terminated child processes. In the event that a catastrophic error occurs, the server can shutdown gracefully rather than leaving a running process in an inoperable state.

Reliability and long term usage of the server have been key design priorities of the thesis. Good design principles and error handling procedures have been observed to insure the server is able to provide uninterrupted service for long periods of time.

4.4 The Cryptographic System Design

The goal of providing our system with a secure communication channel that offers authentication, data integrity, data confidentiality and access control capabilities can be best achieved through the combined use of symmetric, asymmetric and scrambling technologies. We have chosen to use an asymmetric encryption scheme because we need to authenticate, determine access control for each user that wants to access the server and offer a method of conducting transactions over a secure channel. All these goals can be attained through the synergistic use of the RSA asymmetric key algorithm combined with the 3DES symmetric key algorithm. The cryptographic routines used in this thesis are used from an open-source project (http://xyssl.org/) that allows for the encryption code to be encapsulated in our code. This avoids unwanted interdependencies with various other cryptographic libraries available on the market. XySSL is licensed under the terms of the GNU Lesser General Public License (http://xyssl.org/code/LGPL.txt), which allows end-users to link XySSL with either GPL or proprietary programs.

4.4.1 RSA

RSA was one of the first public-key schemes developed. It was invented by Ron Rivest, Adi Shamir and Len Adleman at MIT in 1977 [RSA]. The algorithm is named after their

last names' initials. Ever since the RSA scheme has been the most widely used and implemented scheme in commercial public-key encryption applications. RSA is a block cipher with both the plaintext and ciphertext being integers between 0 and some specific n-1. The scheme involves a public key and a private key. The public key can be and should be known by everyone and is useful for encrypting messages. The private key is supposed to be secret and be kept by the owner of the pair. A message encrypted with the public-key can only be decrypted by the private key. The process works both ways: a message encrypted with the private key can only be decrypted with the corresponding public key. A message encrypted with one of the keys can not be decrypted by the same key.

The security of the RSA algorithm is based on two mathematical problems: the factoring of large integers and the RSA problem. The factoring of large integers is the process of breaking down a number in a pair of its divisors so that when multiplied these divisors yield the same number. With large numbers there is no known algorithm that can determine the two divisors in a polynomial amount of time. The hardest iteration of the problem happens when the two numbers that are multiplied are specifically chosen to be random and prime with lengths being approximately equal in size. The RSA problem is defined as the task of taking *e*th roots modulo a composite *n*: recovering a value *m* such that $c=m^e$ mod n, where (e, n) is the RSA public key and c is the RSA ciphertext. No method to compute *m* has been found yet but there is no proof that such a method exists or that it doesn't exist. This problem is known, but we decided to use the RSA algorithm until the problem has a solution and the RSA algorithm is broken.

Another security problem associated with RSA is the fact that for smaller values for the message M, such as 0 or 1, the ciphertext C is always 0 or 1. The actual encryption or decryption routines use a padding scheme that assures the value of M goes far beyond the small numbers in the ASCII table.

Also for smaller values for e, such as 3, the largest value for an ASCII only M message is 255 and the C is 255^3. This problem is known as the "Small encryption exponent e" [MENEZEZ]. If an entity wants to send the same message M to three entities whose public moduli are n_1, n_2 and n_3 and whose encryption exponent is e=3, then the entity sends $c_i = m^3 \bmod n_i, i = 1,2,3$. Because these moduli are most likely pair wise relatively prime, the attackers are able build an equations system and find an $x = m^3$ and actually by calculating the integer root cube, the original message is recovered. The e value of 65537 $= 2^{16} + 1$ is usually used in RSA implementations as this e value has the advantage that the M message is not usually sent to 65537 recipients. A randomly generated padding attached to any message that is encoded using a small value for e is very helpful to prevent against such an attack.

The RSA implementation uses various techniques to increase the efficiency of computing large integers needed by the RSA algorithm. The Chinese Remainder Theorem for large integers is a method of improving the speed of encryption/decryption using the private key. The amount of computation needed to calculate the original message M is decreased if the modulo n operation is not performed on n, but separated into two modulo

operations performed on p and q according to the Chinese Remainder Theorem (see Appendix A).

By using of the Garner algorithm to implement the Chinese Remainder Theorem, the running time of the RSA encryption/decryption using the private key should be four times faster than the classic method. More information regarding the RSA algorithm and the Chinese Remainder Theorem is provided in Appendix A.

4.4.2 DES

DES (Data Encryption Standard) [3DES] is one of the most widely used symmetric key encryption schemes after its adoption in 1977 by the National Bureau of Standards as a federal information processing standard.

DES is a block cipher that processes plaintext blocks of n = 64 bits producing ciphertext blocks of 64 bits by transforming the plaintext through a series of complicated operations into a ciphertext. The effective size of the key used by DES is 56 bits long. The actual key is specified a 64 bit number, 8 bits (bits index 8, 16, 24… 64) out of which are used as parity bits – being actually discarded - reducing the effective size of the key to 56 bits. The plaintext is passed successively through 16 identical stages of processing called rounds plus an initial permutation (IP) and a final permutation (FP) which effectively undo themselves. Before each round the plaintext block is divided into two 32 bit halves which are processed alternately. This structure allows for the same algorithm to be used both to encrypt and decrypt a message hence the symmetrical scheme name.

DES has two types of security concerns associated with it: one that relates to the algorithm itself as being a freely available algorithm, the second relates to the length of the key of 56 bits.

The first concern can not be proven wrong, but in the years since DES has been available to the public, numerous attempts have been made to break it using one of the general types of attacks: ciphertext-only where the cryptanalyst only has access to some ciphertext snippets, known-plaintext, when plaintext-ciphertext pairs are available and chosen-plaintext when the cryptanalyst has access to ciphertext that are generated based on plaintext messages known to the attacker. Due to the attention that the algorithm has been getting, it is safe to assume that is rather hard to break it until someone finds a fundamental flaw.

The key length though is much more serious potential weakness due to its short size of only 56 bits which allows for only 2^{56} possible keys; making an exhaustive search of the key space completely possible with the high performance multi-processor machines available today. Also, if the attacker attempts a key-space exhaustive search (brute force attack), the attacker must have a method of deciding if a specific message is actual text or not. As such a scrambling method provides the next level of security against attempts to guess the plaintext from an exhaustive search approach.

In order to avoid this key-length security concern, but still take advantage of the speed and long-time exposure of the actual algorithm to the public at large, we have decided to use the 3DES (Triple DES) algorithm. 3DES uses three keys of length 64 bits and runs the DES algorithm three times once for each key.

One security issue of paramount importance for us because we use Triple DES is to make sure that if we had 2 DES keys K1 and K2 there is no other key K3 such that $E_{K3}(x) = E_{K2}(E_{K1}(x))$ which means that multiple encryptions are equivalent to a single encryption. The set of 2^{56} possible key permutations of the key space of DES is not closed under functional composition and a lower bound on the size of the group obtained by composing the set of permutations of the DES functions is 10^{2499}. This result shows that the repeated use of any DES key in an encryption process doesn't run into the problem of being decrypted using an equivalent different key.

4.5 The Symmetric Key Generation Design

The need to generate random pads for frequent similar plaintext, symmetric keys and initial vectors on the fly has led to the development of a procedure that is based on some variable inputs (the time, the process id, a random number, etc.) and it can generate a new random set of bytes.

The procedure requires – as shown in Figure 10 - an input string of characters and a seed number. The string is unique to a specific user (for instance truman@missouri.edu can be

a unique identifier for a user as long as the identity management system of University of Missouri assures that). The seed number is used by the random number generator.

A 16 byte memory buffer is filled with values that use the current time, the length of the string, the current process' ID and the provided seed to compute the required values. Upon completion, the SHA-256 hashing algorithm is used. The hashing algorithm is called 8192 times to compute a message digest over the input string, the memory buffer and its previously computed digest. The resulting digest has 256 bits and is used to retrieve 64 bits for the initial vector and 192 bits for the symmetric key. The DES algorithm requires in its Cipher Block Chaining operation mode (CBC) an initial block of 64 bits to start the encryption or decryption. The symmetric key is used throughout the DES algorithm's rounds to scramble the input plaintexts.

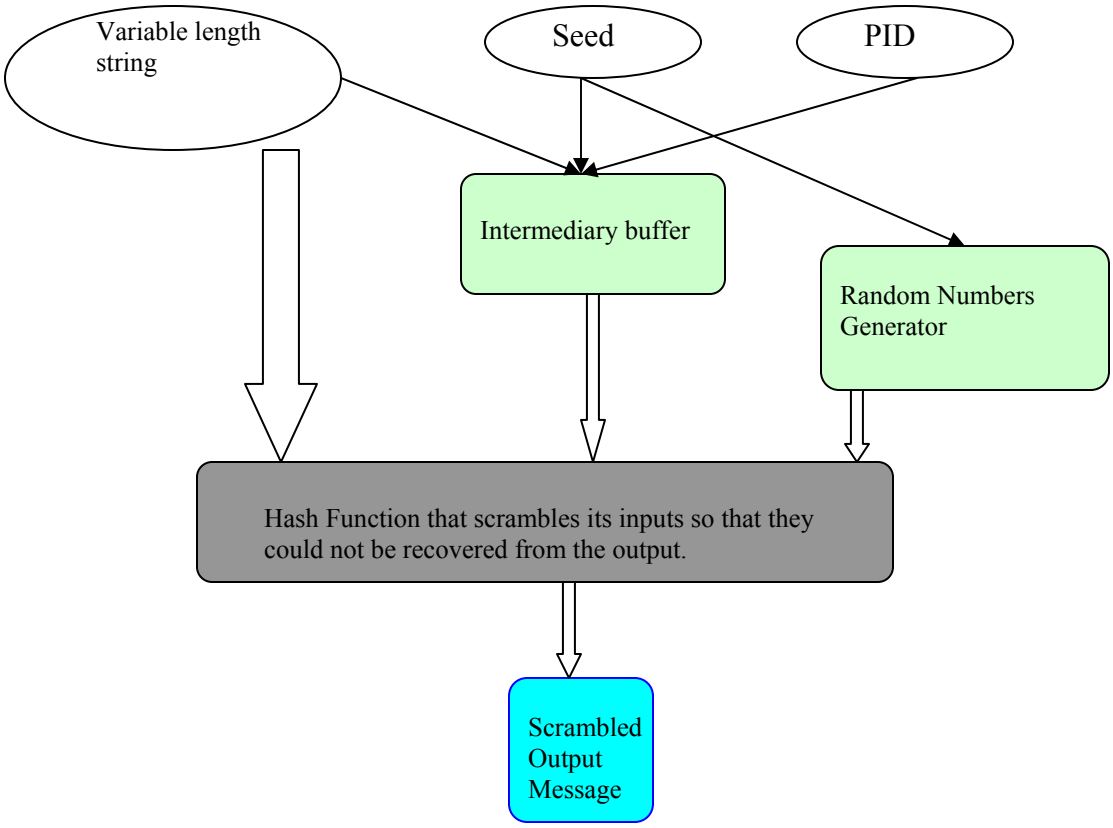

Figure 10. Outline of the scrambling technique

The choice of a hashing algorithm to generate a symmetric key was driven by the fact that it is a one-way function that takes a variable-length message and produces a fixed-size output, called a digest. Its security is based on the fact that it is – with a high degree of probability - computationally infeasible to find the original message from its digest or to find two messages that have the same digest message. Any change in the message will result in a different digest with a high probability. Based on these characteristics of the secure hashing algorithms we have chosen the SHA-256 (Secure Hash Algorithm - 256) algorithm (the size of the digest message is 256 bits) due to its secure nature. The SHA-1

version has been broken in 2005 and the SHA2 version has not been proven insecure until now.

4.6 The Gdbm Database

The structure of the data that needs to be manipulated and the quick access requirement have been of paramount importance in selecting the type of underlying database to use in order to fully support fast data retrieval. Our data is composed of the user name, the institution name, the virtual organization name and an entitlement value as depicted in Figure 11. In order to retrieve these values in a fast manner we have decided to use a lookup table style of database that supports fast retrieval with little overhead. A relational database, such as mySQL or Oracle, was also considered in the process of selecting a database to use in this thesis. For the purposes of developing a prototype entitlements mechanism it was decided to leave a relational database approach to a later stage of development.

Virtual Organization Name	Institution Name	User Name	Entitlement Value

Figure 11. The entitlements database entry structure

Gdbm (GNU database management) is the GNU implementation of the original UNIX dbm library widely used in a variety of applications. Gdbm is a light-weight alternative to

a full-blown relational database because it doesn't have all the features associated with a relational database. Gdbm is a file system-based hash table that pairs a key with data and allows for the data to be retrieved by means of the keys. The keys are not stored in alphabetical order are randomly distributed throughout the database. The database is hash table which has the property of a hash to make high speed location of a desired record in the table possible.

The database is backed by a separate file on disk that persistently stores the keys and their data. We wanted to be sure that any type of data inconsistencies in the database due to many processes writing and reading data in the database simultaneously can happen. Each time one wants to use the database, they need to open it. The database can be open in two specific modes: reader or writer. In order to avoid race conditions (where more than one process tries to read and modify the data at the same time leading to inconsistencies of the database) in the database, gdbm allows the database's file to be opened by as many readers (a process that only reads data from a file) as wanted, but only by one writer (a process that reads and writes data to and from a file). If a database file is already opened by a writer, then a reader isn't able to open it until the writer has finished their activity. The writer mode allows for addition of new records into the database, while the reader mode only allows for searches.

The gdbm database uses the following structure to describe a key and its corresponding data:

```
typedef struct {
   char *dptr;
   int  dsize;
} datum;
```

In our design – in order to support fast data retrieval – the key of each record is formed from the user name, the institutional name, the virtual organization name and an entitlement following the format:

<virtual organization>@<institution name>@<user name>@<entitlement>.

For instance,

greatplains.net@missouri.edu@ciordas@urn:mace:greatplains.net:repository

The data corresponding to each key holds the entitlement field of the key. This structure allows for a decision regarding the existence of a user's credentials to be reached in only one operation. A previous design stored all the entitlements of a specific user name in the data field of the record. After running a benchmark test using the two designs, it turned out that adding and deleting an entitlement had to deal with an array of strings that needed to be shifted and reformatted every time a delete or an add operation was performed. The time penalty in these cases was far greater than experienced with our current design. Also the size of the database file increased exponentially in the second

design. As such, the original design choice is not suitable for the purpose of our thesis and will not be used.

The operations offered by the database are add a new record, delete an old record and retrieve/fetch an already existing record. Due to lookup table nature of our database, the structure, the operations and the speed and simplicity offered by gdbm make it the best choice for our application.

4.7 Message Scrambling

In order to further avoid attempts of recognizing and using the information contained in the exchanged messages between the client application and the server application, a method of scrambling the message before being encrypted seemed feasible. This can be done by specifying a special code for the numbers exchanged, or using a foreign language to obfuscate the messages, or using a cipher such as the Caesar cipher. The choice of a polyalphabetic cipher was made because of the underlying properties of making frequency analysis more difficult and because of its speed of execution.

A polyalphabetic cipher is a cipher based on substitution and using multiple substitution alphabets. One of the most well-known ciphers is the Vigenere cipher that substitutes each character in a message with a corresponding character in its inner table according to the characters in a keyword. The inner table is composed of 128 rows according to the number of ASCII characters and each row is comprised of the entire ASCII table shifted left one position. The encryption process uses a key that is repeated as many times as

needed in order to cover the length of the message to be encrypted. Subsequently, every character in the message has a corresponding character in the key, and using these two characters in the table as the row and column indicators, a third character emerges that is used as the cipher for that particular character from the original message. The decryption follows along the same path, using the ciphertext as the starting point and the same shared key in the table to decrypt the message.

One of the most important characteristics of the Vigenere substitution cipher is that it doesn't allow for frequency analysis to be performed easily on the ciphertext. Frequency analysis is the process of counting the appearances of each character in the ciphertext and trying to substitute them with the most used characters in the English language. For instance the letter E is one of the most used letters in any message, and as such it can be easily spotted in a ciphertext that encodes each character with the same code character. The very best quality of the Vigenere cipher is also its weakness, because as the length of the key gets smaller, the chance of finding repeating patterns in the ciphertext increases, leading to frequency analysis that easily yield the key value.

By itself, the Vigenere cipher is not secure, but used in combination with a symmetrical or asymmetrical scheme, it provides a new level of security, especially if the length of the key is long enough not to allow for easy interpretations of the resulting ciphertext.

4.8 The Authentication Scheme Design

The users and service providers who want to access the services of our application need to be authenticated and authorized prior to any communication with our system can take place.

For the service providers our authentication and authorization design encompasses an authentication scheme based on a public-private key scheme. Each service provider has a private key of their own generated by the entitlements server which holds the public key of that service provider. Also each user has access to the public key of the entitlements server.

With such an infrastructure in place, once the service provider begins the communication, it uses the server's public key to encrypt its authentication message and uses its own private key to encrypt the resulted ciphertext. This allows for non-repudiation, data integrity and allows for each party to verify that only the intended recipient has all the necessary keys to actually decipher and use the contents of the authentication message.

For administrative users who want to use the entitlements server to perform updates and lookups, they firstly need to be authenticated by their identity provider using the Shibboleth mechanisms and only upon successful completion of the authentication step, they are authorized by the entitlements server using the already established secure channel with the service provider.

The scheme was chosen due to relatively small number of service providers that need access to the entitlements server at an institutional level and also because of the security provided by the encryption scheme.

5. The Client Implementation

5.1 Client Application Functionality Outline

The client application is primarily responsible for servicing the service provider's requests to the entitlement server. The client application needs to be able to connect to the server, it needs to be able to follow the server's communication protocol and communicate using the server's message types. The client application needs to be able to offer its users with an interface that allows them to query the server, to add or delete records to the server's database and also to terminate a session.

Firstly, the client application starts by verifying its arguments in order to decide if the basic information required is provided by the user. The format of the arguments is comprised of the DNS name of the server to be contacted, the service provider username, its home institution, its virtual organization and a numeric value of 10 for SP_SETUP operation, 20 for SP_LOOKUP operation or 30 for an SP_USE operation. Each one of these arguments is strictly verified and the values are saved for further processing. These numerical values have been chosen in order not to conflict with the numeric values associated with the administrative user commands (add – 1, delete – 2, lookup – 3, display entitlements – 4, display users – 5, display all – 6, and logout - 7).

Secondly, a communication channel is established with the server before any communication is possible. Using the service provider's username and verified server name and its previously-known port, the client application establishes a TCP connection

with the server. It tries to reach the server for a number of seconds, giving up upon the elapse of a previously defined amount of time without a response. If the server is busy servicing other requests or is not functioning and it doesn't answer the connection request for a number of seconds, the client ceases to try to connect and an attempt can be made at a later time. Upon successful completion, a communication channel with the server is opened and ready to be used by the ensuing messages.

In case of an SP_SETUP operation, the client application waits for the entitlement server's authentication and authorization decision. If the authentication process is successful, the client application receives the entitlements server's session ID comprised of one 3DES symmetric key and an initial vector (IV) that are both used in the following secure communications 3DES encryption/decryption. The secure communication channel is established using the 3DES symmetric key Chain Block Cipher (CBC) encryption and decryption technology due to its speed in processing a message and its cryptanalysis resistance strength. The symmetric key is saved on the local machine so that the client application can use it in future communication with the entitlement server.

In case of an SP_LOOKUP operation, the client receives the session status message from the server. If the session is still valid, the client sends a query and displays the server's response. In case of an SP_USE operation, the client receives the session status for both the service provider and the administrative user that sends commands. If both have non-expired sessions, the client sends the actual command (add, delete, lookup, display or

logout) followed by the entitlement to be used by the command. The server responds with a status message that is passed on by the client application.

Upon completion of the communication, the client application closes the TCP connection used to communicate with the server, frees any allocated memory and finishes gracefully.

5.1.1 The Client Application's Interfaces

The client application can be called from the command line and passed the necessary arguments and it contacts the entitlements server and performs its job. Also, the client application can be used in a web-enabled environment where the client application is started by the web server while executing some PHP scripts.

For instance, when using the command line client application, the service provider needs to have a copy of the client application and the corresponding public private key necessary to communicate with the entitlements server. The service provider's administrator can use the client application (while being logged into the service provider's computing environment) to issue SP_SETUP, SP_LOOKUP or SP_USE commands. The format of the commands messages are described in the next section. As of this prototype, the service providers can run the client application and the regular users who want to get access to a resource protected by the service provider need to use the web interface of the requested computing resource. A future enhancement of this prototype will take into account the needs of users who require access to a shared resource using a command line interface. The command line interface will need to

communicate with the identity provider to authenticate its user, with the service provider in order to authorize its user and finally with the actual resource. Our client application will find its place as a component of the service provider's command line interface.

In the web implementation, in order to access the administrative capabilities of the entitlements server, an administrative user needs to connect to the virtual organization's webpage and be authenticated by his home institution's identity provider. Upon successful completion of the Shibboleth authentication, the browser redirects the user to the authorization page where the user needs to identify the name of the virtual organization that he wants to be granted administrative rights to. The web server environment provides the user's principal name (e.g., ciordas@missouri.edu) as provided by the identity provider.

Using this data, the authorization page builds a client application command (SP_LOOKUP) and sends it to the entitlements server. If the user is authorized by the entitlements server, he is presented with an administrative page that allows the user to make requests according to the level of privilege allowed by the entitlements server by issuing SP_USE commands to the server through the client application.

5.2 The Authentication and the Command Messages

An authentication message needs to incorporate the most useful data about a user (a service provider, an administrative user or a regular user) avoiding any privacy issues. The user name is used to uniquely identify the user in the server's database along with its

home institution name and its virtual organization. Besides these pieces of information, the authentication message also contains the type of operation requested (SP_SETUP, SP_LOOKUP or SP_USE). These pieces of information are gathered with the format:

<user name> <institution name> <virtual organization name> <operation type>.

The command messages are the messages sent as a result of the commands issued by the service provider or the administrative user. The structure of the commands is decided by the specific code of the command and its respective parameters and is used to be passed in as arguments to the client application.

Based on the format of the command, the server retrieves the required information and performs its requested action and returns a result.

6. Integration with Shibboleth

The fine-grained authorization service offered by the server in this thesis is designed to work somewhere in the physical space of an institutional part of a virtual organization serviced by the entitlements server. As such, the authentication and authorization processes take place in a distributed fashion with the identity provider in charge of the authentication and the service provider in charge of the authorization.

By having the entitlement management administered by the entitlements server on behalf of a virtual organization, we avoid many of the privacy and security policy problems that make the current design of Shibboleth harder to implement by institutions. Service providers call upon the entitlements server to answer queries and respond to administrative requests by authorized administrative users. Whenever a user is authorized to use a specific service, an administrative user from the institution that offers the shared service updates the entitlements database or delegates the responsibility. The identity providers do not need to be concerned with the managing the virtual organizational related entitlements as they do not care or need to know what users are entitled to use what resource in a virtual organization that the current organization is a member.

The entitlements server and the client application described in the thesis outline a functional and easily extensible framework that allows for the service provider to implement the logic of access control as they see fit and also allows for the administrative users of the entitlements database to have easy access to the entitlements server either

through the provided command-line interface or through a web-based interface that uses the client application.

The service provider may choose to have an access control policy that uses a hierarchy of entitlements such as having one entitlement that is a super-set of other entitlements. For instance, the urn:mace:greatplains.net:biogrid entitlement could be a super-entitlement for urn:mace:greatplains.net:missouri.edu:biogrid, urn:mace:greatplains.net:uark.edu:biogrid and urn:mace:greatplains.net.ku.edu:biogrid. If a user has only one of the three previous entitlements, the user has access only to the specified university resource; otherwise, if the user has the initial super-entitlement, access to any of the three institutions's biogrid facilities would be granted. This provides flexibility of using definable entitlements.

The access control logic that deals with creating boolean expressions that decide if a user is authorized to use a resource can be located in the Shibboleth's resource manager and as such the resource manager creates a boolean expression and interrogates the fine-grained authorization entitlements server for each part of the expression, deriving the result after a series of interrogations. Another approach is to have the access control logic inside the fine-grained authorization server and by passing the server a boolean expression and the identity information of a user, the server can be able to derive the needed access control decision.

Our implementation chooses to keep the access control logic out of the server logic in order to allow for flexibility in dealing with further needed changes in the Shibboleth's implementation. As the nature of the Great Plains Network virtual organization changes, so the nature of the access control policies used by the resource manager might change. If the need arises to implement the access control logic into the fine-grained authorization server, the extensible design of the server allow for a boolean expression in an agreed-upon format to be passed along and the decision to be made, returning the result to the service provider's resource manager.

In order to demonstrate the integration of our client application with the Shibboleth environment, a PHP script is written to authorize a user's access to the Great Plains Network access web page – https://osprey.rnet.missouri.edu/authtest/PHP/GPN.

The index.php script contains a function check_entitlement that verifies the existence of the provided entitlement value:

```
function check_entitlement($EntlServerName, $SPPrincipalName, $SPVO,
$PrincipalName, $VO, $Entitlement)
{
//Execute an SP_LOOKUP(=20) command to lookup the needed entitlement
      exec("/var/www/html/authtest/PHP/db_client 20 $EntlServerName
$SPPrincipalName $SPVO $PrincipalName $VO $Entitlement", $b);

      //If the SP session is expired or never established, then
      //establish one using operation SP_SETUP(=10)
      if($b[1] == "SP_SESSION_EXPIRED" ||
      $b[1] == "Error_open_\"symmetric.key\"_file")
      {
            //Execute a SP_SETUP(=10) command to establish a new
            //session
            exec("/var/www/html/authtest/PHP/db_client 10
            $EntlServerName $SPPrincipalName $SPVO", $a);

            //If the session is established correctly, reissue the
            //authorization command
```

```
            if($a[1] == "SP_SETUP_SUCCESS")
            {
            exec("/var/www/html/authtest/PHP/db_client 20
$EntlServerName $SPPrincipalName $SPVO $PrincipalName $VO
$Entitlement", $c);
            }
            else
            {
                  if($a[1] == "SP_AUTHENTICATION_FAILED")
                        print "<B>Unauthorized access:</B> The SP
doesn't have the proper entitlement to access this page.<BR><BR>\n";
                        exit;
            }
      }

      if(($c[1] == "SP_SESSION_OK" && $c[2] ==
"USER_ENTITLEMENT_LOOKUP_SUCCEEDED")
      || ($b[1] == "SP_SESSION_OK" && $b[2] ==
"USER_ENTITLEMENT_LOOKUP_SUCCEEDED"))
            return 1;
      else
            return 0;

      return 0;
}
```

The function receives the parameters $EntlServerName that represents the entitlements server address, $SPPrincipalName represents the service provider user name and institution, $SPVO represents the service provider virtual organziation, $PrincipalName represents the user name and institution, $VO represents the virtual organization, and $Entitlement represents the searched entitlement value.

A SP_LOOKUP command is issued through the client application using the function's arguments. If the service provider's session is already expired or it has never been established before, the client application is called to issue a SP_SETUP command in order to establish a secure communication channel with the entitlements server. Otherwise, if the lookup succeeded the function returns true, otherwise it returns false.

Figure 12 depicts the favorable authorization decision reached for the user ciordas@missouri.edu who has the entitlement "urn:mace:greatplains.net:repository" in the entitlements server upon completing the Shibboleth authentication.

Figure 12. Authorization using the Entitlements Server.

Also, the administrative user can obtain access to the entitlements server through a web interface. Figure 13 depicts the administrative page that an administrative user can reach upon being properly authenticated by their identity provider and authorized by the entitlements server.

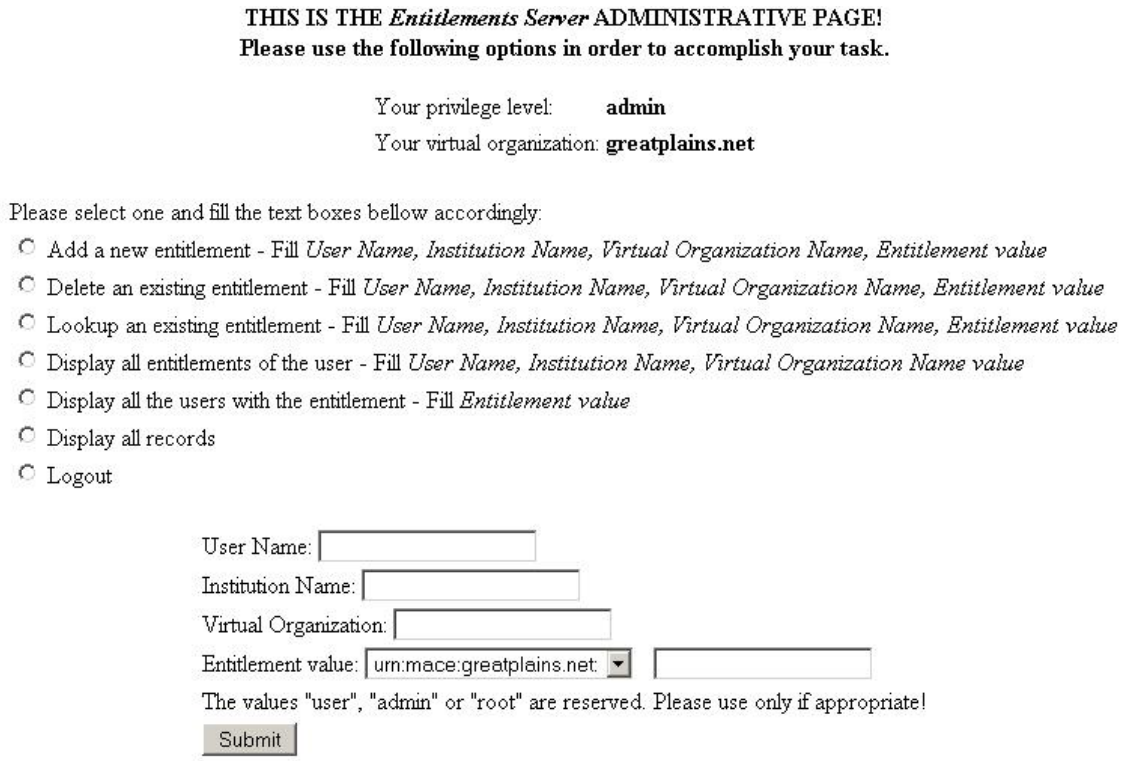

Figure 13. Entitlements Server's administrative page

From this page, the administrative user has the option to add, delete, and lookup or display various users' entitlements.

7. Conclusion

The last few years have experienced a steady growth in research institutions showing interest in developing research projects that involve more than one institution's computing resources. This movement towards sharing of resources and capabilities from one university to the other is the basis for the Great Plains Network Consortium (GPN) of universities from the Midwest that decided to come together and form an organized manner of sharing resources under the protection of a common set of policies. The goal of the Great Plains Network, as well as of Internet 2, is to use standards-based middleware to provide member institutions with a collaborative research environment free of inter-operability problems, and at the same time an easy to use and secure environment for scientists to be productive. This environment should allow entitled researchers from any member institution to use any shared resource of the virtual organization by means of any appropriate interface (e.g., web browser, command-line application).

Shibboleth was the first choice of infrastructure to be used to create a collaborative inter-institutional research environment. Shibboleth provides a virtual organization the possibility to share resources across multiple institutions without having to create physical user login ids on each of the systems that are shared. A user needs to be registered with an institution prior to using any of the shared resources. The user can use his or her authentication credentials from their home institution to gain access to all the shared resources that the user is entitled to use. This has the advantage of lowering the

administrative overhead of maintaining ids on all of the available machines being shared. Shibboleth was the perfect framework to have this vision come to life due to its design based on open standards and usability. This methodology also benefits from the evolving "middleware" standards being developed under the auspices of the Internet 2 organization and the National Science Foundation. All the GPN member institutions are also members of the Internet 2. Initiatives such as this one are an important part of the Internet 2 mission to facilitate advanced network-based applications to support research and education across institutions of higher education in the United States.

The creation of a collaborative research environment that spans multiple institutions each with their own policies and conventions is a real challenge. Some of the problems that have been encountered in practice with Shibboleth deal with the distributed authentication and authorization mechanisms. In the standard Shibboleth architecture, the identity provider (the user's home institution) is in charge of authenticating the user and also of storing all the entitlements that the service provider (the shared resource) is basing their access control decisions. Business and user privacy policies make it difficult to deal with the storage and management of all the entitlements in the identity provider. One problem pertains to the security risks involved in allowing external entities to manage entitlements in the identity management system of an institution. Another problem emerges from the identity providers' usage of the eduPerson object class to store the attributes and entitlements that pertain to a user. The eduPerson object class does not support significant modifications in its format to allow for virtual organization-defined entitlements to be stored in its structure. The available approaches such as Signet and

Grouper use only eduPerson objects to store attributes. Due to the limitations of the eduPerson object class they are unable to implement the stated goal of an inter-institutional collaborative research environment.

This thesis addresses some of the issues that have slowed the adoption of Shibboleth-based authentication and authorization systems in deploying fully collaborative research environments. In order to allow for fine-grained authorization at the virtual organization level, there is a need to define, manage and use virtual organization entitlements independently of any institution or participating company. These entitlements do not refer to any particular user or institution. They encompass the idea of a shared resource that needs to be made available to any entitled entity from any member organization. In order to attain the goal of a fully collaborative inter-institutional research environment, the definition, management, and usage of the virtual organization-defined entitlements are maintained separately from the identity provider – the entitlements repository. This repository is available to any service provider from any virtual organization to assert entitlement values and to any administrative user to query and update the repository. In our view, the identity provider is in charge of authenticating the users, the service provider is in charge of the authorization decisions, and the entitlements repository is in charge of defining, managing and providing access for queries and updates to be run by the stored entitlements service. The identity provider, the service provider and the entitlements service jointly provide for creating a secure and robust collaboration environment for use by virtual organizations.

The need to have virtual organization-defined entitlements that pertain only to a virtual organization without any relationship to the entitlements that describe an institutional user led to the idea of creating an entitlements repository that manages all these defined entitlements. For instance, a user who wants to access a resource can have the "faculty member" or "registered in CS 1001" attributes stored in its eduPerson object, but in order to use a resource that requires membership in the "biogrid" group denoted by the "urn:mace:greatplains.net:biogrid" entitlement, they need to have an entry into a repository somewhere that links their identity to the mentioned entitlement.

The creation of an entitlements repository provides for the separation of the authentication and authorization steps, allowing for finer-grained access control decisions to be made by the service provider. The identity provider's identity management system doesn't want or need to be in charge of the management of any virtual organization-related entitlements for its members. The virtual organization wants control of the virtual organization-related entitlements. Our entitlements repository provides a complement to the identity provider's attributes authority in terms that it allows extending the reach of access control decisions to entitlements that cannot be defined within the confines of the eduPerson object.

Our prototype implementation provides for a secure way of storing and managing virtual organization defined entitlements. The separation of the virtual organization defined entitlements from the identity provider, allows for a wider adoption of the Shibboleth-enabled authentication and authorization schemes at the virtual organization level. We

hope that by eliminating the logistical and privacy issues associated with storing and managing virtual organization defined entitlements at the institutional level, many institutions will adopt the technology and join a growing number of research groups that share resources and bring new inventions to life.

The proposed entitlements repository prototype needs to be enhanced in order to become a robust and widely used technology. Some proposed enhancements are needed in order to take the proposed technology from the prototype state to the production state. A relational database management system needs to be employed in order to provide for a full fledged database system that can support the needs of a large set of virtual organizations. A user interface using a command line, besides the provided web-based user interface, needs to be developed in order to leverage the need for this type of interface. The mechanism needs to be integrated with the evolving Signet and Grouper developments in order to make it easily accessible to the academic research community. Lastly, a customization package needs to be developed in order to facilitate the creation of a new virtual organization with needs for entitlements services.

This thesis and prototype implementation opens the opportunity to make it easier to facilitate collaboration between any groups of research institutions. The thesis is a proof of concept that hopes to enhance the opportunities for multi-institutional collaborations. While its limitations might prevent wide adoption of the prototype, we hope that the suggested enhancements and proposed ideas will be integrated in the future iterations of the Internet 2 middleware infrastructure.

Bibliography

1. [MENEZEZ] Alfred J. Menezez, Paul C van Oorschot, Scott A. Vanstone, "Handbook of Applied Cryptography", CRC Press Ltd., August 2001
2. [UNIXNET] W. Richard Stevens, "UNIX Network Programming Volume 1", Prentice Hall, 1998
3. [STALL] William Stallings, "Network Security Essentials: Applications and Standards", 3rd Edition, Prentice Hall, 2007
4. [SHIBINTRO] Amy Apon, Kurt Landrus, "Introduction to Shibboleth", Computer Science and Computer Engineering Department, University of Arkansas, Fayetteville, AR 72701
5. [SCPE] Amy Apon, Gregory Monaco, Gordon K. Springer, "The Great Plains Network (GPN) Middleware Test Bed", Scalable Computing: Practice and Experience, Volume 7, Number 3, pp. 95 – 108, 2006
6. [AUTHN] http://en.wikipedia.org/wiki/Authentication
7. [AUTHZ] http://en.wikipedia.org/wiki/Authorization
8. [PKCERT] http://en.wikipedia.org/wiki/Public_key_certificate
9. [IDM] http://en.wikipedia.org/wiki/Identity_management
10. [RSA] R. Rivest, A. Shamir, L. Adleman., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, Vol. 21 (2), pp.120–126. 1978.
11. [3DES] http://en.wikipedia.org/wiki/Triple_DES
12. [VIGENERE] http://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher
13. [CRT] http://www.cut-the-knot.org/blue/chinese.shtml
14. [FERMAT] http://en.wikipedia.org/wiki/Fermat%27s_little_theorem
15. [SHIBB] http://shibboleth.internet2.edu/
16. [WIKISHIB]http://en.wikipedia.org/wiki/Shibboleth
17. [MACE] http://middleware.internet2.edu/MACE/
18. [GETTES]http://www.educause.edu/LibraryDetailPage/666?Redirect=True&ID=EAF0429
19. [HAZELTON]http://www.educause.edu/Elements/Attachments/netatedu/pki/eduperson/hazelton.ppt
20. [GPNDOCS]http://archie.csce.uark.edu/gpn/publications.html
21. [MACEGPN]http://crick.rnet.missouri.edu/mace-gpn/
22. [EDUPERSON]http://www.educause.edu/eduperson/
23. [SAML]http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security
24. [XML] http://www.w3.org/XML/
25. [GPN]http://www.greatplains.net/
26. [MACE] http://middleware.internet2.edu/MACE/
27. [INT2] http://www.internet2.edu/

Appendix A - RSA Implementation

The process of generating the RSA public and private keys requires the following these steps:

1. Choose two random prime numbers p and q.
2. Compute n = p*q.
3. Compute *the totient* φ (n) = (p-1)*(q-1). *A totient* of an integer n is defined to be the number of positive integers less than or equal to n that are coprime to n.
4. Choose an integer e such that 1 < e < φ (n) and gcd (e, φ (n)) = 1 (gcd = greatest common divisor).
5. Calculate d such that d*e mod φ (n) = 1.

The public key is the pair (e, n) and the private key is the pair (d, n). The encryption of a message M (M < n) has the following format: $C = M^e \pmod n$ where the message M is raised to the power of e and its modulo n value is computes and represents the ciphertext C. The decryption of a ciphertext C has the following format: $M = C^d \pmod n$ where the ciphertext C is raised to the power of d and its modulo n value represents the original message M.

Fermat's Little Theorem: If p is a prime number, and a is an integer such that (a, p) = 1, then

$a^{p-1} = 1 (\mathrm{mod}\ p) \Leftrightarrow a^{p-1} \bmod p = 1$. [FERMAT]

The Chinese Remainder Theorem: If the integers $n_1, n_2, ..., n_k$ are pair wise relatively prime, then the system of simultaneous congruencies

$$x \equiv a_1 (\mathrm{mod}\, n_1)$$
$$x \equiv a_2 (\mathrm{mod}\, n_2)$$
.
.
.
$$x \equiv a_k (\mathrm{mod}\, n_k)$$

Has a unique solution modulo $n_1, n_2, ..., n_k$.

Garner's algorithm to find the solution for the Chinese Remainder Theorem:

Input: a positive integer $M = \prod_{i=1}^{t} m_i > 1, with \gcd(m_i, m_j) = 1 \; for\, all\, i \neq j$ and a modular representation $v(x) = (v_1, v_2, ..., v_t) of\ x\ for\ m_i$.

Output: the integer x in radix b representation.

1. For i from 2 to t do:

    1.1 $C_i \leftarrow 1$.

    1.2 For j from 1 to (i-1) do:

$$u \leftarrow m_j^{-1} \bmod m_i$$
$$C_i \leftarrow u * C_i \bmod m_i$$

2. $u \leftarrow v_1, x \leftarrow u$

3. For i from 2 to t do:

$$u \leftarrow (v_i - x) C_i \bmod m_i, x \leftarrow x + u * \coprod_{j=1}^{i-1} m_j$$

4. Return x. [MENEZEZ]

The Garner algorithm has a special case for RSA decryption modulo n=p*q, where $m_1 = p$ and $m_2 = q$ are distinct prime numbers. Step 1 computes $C_2 = p^{-1} \bmod q$. Step 3 computes $u = (v_2 - v_1)C_2 \bmod q \; and \; x = v_1 + up$.

Considering C the ciphertext and M the decrypted message, the RSA decryption algorithm computes:

$M = C^d \bmod n, where \; n = p * q$. We could also compute $v_1 = C^d \bmod p$ and $v_2 = C^d \bmod q$. According to the Chinese Remainder Theorem, we can be sure there is a solution for these two equations in the form of $C^d \bmod(p * q) = C^d \bmod n$.

Due to the Fermat Little Theorem $a^{p-1} \bmod p = 1$ we can rewrite the above two equations as $v_1 = C^{d \bmod (p-1)} \bmod p$ and $v_2 = C^{d \bmod (q-1)} \bmod q$. Using the special case of the Garner algorithm we can find a solution to the above two equations $C^d \bmod n = v_1 + up$.

Appendix B – DES Implementation

The encryption of a block of 64 bits proceeds in 16 rounds. The key is used to generate sixteen 48 bits long subkeys used once for each of the 16 rounds. Each round is composed of 8 fixed 6-to-4 bits substitution mapping boxes (S-boxes) that get the input and produce the round's ciphertext using the provided message and the subkey. The plaintext is divided into two 32 bits long halves denoted L_0 and R_0. Reach round is functionally equivalent to any of the previous or following rounds taking 32 bits of input L_{i-1} and R_{i-1} from the previous round (i-1) and produce 32 bits outputs for the current round (i) L_i and R_i where i = 1 … 16 as follows:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \text{ XOR } f(R_{i-1}, K_i); \text{ where } f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \text{ XOR } K_i)$$

E and P are fixed expansion permutations mapping R_{i-1} from 32 bits to 48 bits.

Triple DES encryption works as follows:

$$C = E_{K3}[D_{K2}[E_{K1}[P]]]$$

where C – ciphertext, P – plaintext, $E_K[X]$ – encryption of message X using the key K and $D_K[X]$ – decrypting message X with the key K.

The decryption follows the steps with the keys in the reverse order:

$P = D_{K1}[E_{K2}[D_{K3}[C]]]$

where C – ciphertext, P – plaintext, $E_K[X]$ – encryption of message X using the key K and $D_K[X]$ – decrypting message X with the key K.

Appendix C – Entitlements Server Description

Name:

db_server – runs the Entitlements Server

Synopsis:

db_server [-options]

Options:

-d[1 or any positive value]: This setting turns on the debugging flag allowing to follow the server's execution. The default setting is not to display debug information. The same effect can be obtained by using –d0.

-p<port number>: This setting allows selecting a different port than the default one to listen for client requests. The default port number is 9544. For instance, -p1234 is a valid option.

-l<pathname>: This setting allows the entitlements database to be reconstructed using an input file that contains all the records that the entitlements database will hold. Using this setting, the entitlements database can be restored to a state described by the contents of the file pointed to by the "pathname". The file is comprised of a set of lines, where each line has the format:

userID=<username> institution=<institution name> vo=<VO name> entitlement=<entitlement value>. For instance, “userID=smore institution=Mizzou.edu vo=greatplains.net entitlement=urn:mace:greatplains.net:biosci” is a correct entry line in the file. The file must be reachable by the db_server executable. For instance, -l\home\user\Mydblist is a valid option that will read the entries from the “Mydblist” file and store them in the entitlements database.

-t<time value>: This setting allows changing the amount of time a service provider or administrative user session lasts. The “time value” is expressed in seconds. For instance, -t300 is a correct option, instructing the server to use a 5 minutes or 300 seconds session window.

-k<file name>: This setting allows changing the name of the entitlement server’s private key file name. The default name for the server’s private key is “private_key.cert”. Accordingly, all client applications need to have access to the server’s public key file with the default name “public_key.cert. For instance, -kserver.private is a correct option instructing the server to use the file named “server.private” as the file to read its private key from.

-r<dirname>: This setting allows specifying a dataabse files directory. The default server behavior is to search for the key files in the current directory.

-s<dirname>: This setting allows specifying a key files directory. The default server behavior is to search for the key files in the current directory.

Description:

The entitlements server is an autonomous service that manages entitlement values for any virtual organization or federation that decides to use its services. The server answers queries sent by service providers and executes commands issued by administrative users in order to update or query the server's entitlements database.

The entitlements server communicates with client applications located at the service providers. The communication takes place over a secure communication channel that is established by the two parties (the server and the client). In order to establish a secure channel, the server and the client are sharing a set of private-public keys. In its running directory, the server has its own private key file and a file for the public key of each one of its service providers. Similarly, each client application has a copy of the server's public key file and their own private key file. These public-private keys are used to inform the server about the credentials of the service provider requesting service and also to exchange a symmetric key that will be used in all subsequent communication. The symmetric key that is generated by the server and sent to the client represents the guarantee of the secure communication channel.

Each communication with the client starts by the server receiving an encrypted message with its own public key that contains the client's public key file name. This name is used to decrypt the following message that contains the service provider's credentials and the

requested service option encrypted using both the server's public key and the client's private key.

The server communicates using a protocol that is comprised of three commands: SP_SETUP, SP_LOOKUP and SP_USE whose format will be discussed in the Appendix D.

SP_SETUP is used by the client to establish a secure communication channel with the server and to get authorized by the entitlements server. Once the server receives the aforementioned service provider's credentials, it makes an authorization decision based on the existence of the entitlement "user" associated with the requesting service provider. The authorization response is sent back to the client. In case the service provider is authorized to use the entitlements server, the server generates a symmetric key or session ID that is returned to the client via the public-private key infrastructure already in place. The symmetric key will be stored by both parties and used in subsequent communication. Also a time stamp is stored by the server in order to monitor the time the service provider can use the symmetric key to communicate with the server.

SP_LOOKUP is used by the service provider to send queries to the server. The service provider uses this command to inquire the server about the existence of a user's entitlement in order to make an access control decision. In case an administrative user requires access to the server, the service provider queries the server about the existence of the "admin" entitlement or the "root" entitlement according to the privilege level desired

by the administrative user. If the administrative user is authorized to use the server's services, a time stamp is added to the server's time database in order to keep track of the time – session a user is allowed to communicate with the server.

SP_USE is used by the service provider to send administrative users queries and updates to the entitlements server. The privilege level of the administrative user allows for only certain operations to be performed. A regular administrative user (a user that has an "admin" entitlement in the server's entitlements database) is only allowed to add or delete records that belong to its own institution and virtual organization. A root administrative user (a user that has a "root" entitlement in the server's entitlements database) is allowed to perform any type of alteration to the entitlements database.

Operation:

The executable file (db_server), the server's private key file, the service provider's public key files and the entitlements database file (currently named "Mydb") need to be accessible to the server executable file. The optional file containing a description of the entitlements database that is used with the –l option needs to be also accessible to the server.

The server application creates two files timeDB and symkeyDB that represent the session time and the symmetric key repositories.

The server application can be terminated by issuing a SIG_INT signal.

In order to generate a public-private key pair, the user has the option of using an application called “keygen” that generates the appropriate key pairs for the server application or the client application.

Keygen Synopsis:

./keygen <public_key_file_name> <private_key_file_name> <option:1-server/2-client>

<public_key_file_name> represents the name for the file to store the public key

<private_key_file_name> represents the name for the file to store the private key

<option:1-server/2-client> represnts the type of public-private key pair desired. 1 represents a server key which is 1024 bits in length. 2 represents a client key which is 2048 bits in length.

Keygen Operation:

The keygen program creates the required public and private keys, stores them in the designated files and also performs a dummy test in order to be sure they can be used to encrypt and decrypt messages.

Appendix D – Client Application Description

Name:

db_client – runs the Client Application

Synopsis:

db_client <options according to the type of operation: SP_SETUP, SP_LOOKUP or SP_USE>

Options:

1. If the desired operation is SP_SETUP, the format of the client application's parameters is:

<command number (SP_SETUP=10)>

<Entitlements Server Machine Name>

<Service provider username@Service provider institution name>

<Service provider virtual organization name>

For instance, the following line

"*10* beagle.rnet.missouri.edu

osprey.rnet.missouri.edu@missouri.edu **greatplains.net**"

issues a SP_SETUP (*10*) command to the entitlements server providing the entitlements server's name (beagle.rnet.missouri.edu), the service provider username (***osprey.rnet.missouri.edu@missouri.edu***) and the service provider's virtual organization (**greatplains.net**).

2. If the desired operation is SP_LOOKUP, the format of the client application's parameters is:

<command number (SP_LOOKUP=20)>

<Entitlements Server Machine Name>

<Service provider username@Service provider institution name>

<Service provider virtual organization name>

< username@institution name>

<virtual organization>

<entitlement>

For instance, the following line

"*20* beagle.rnet.missouri.edu

osprey.rnet.missouri.edu@missouri.edu **greatplains.net**

ciordas@missouri.edu greatplains.net **urn:mace:greatplains.net:repository**"

issues a SP_LOOKUP (*20*) command to the entitlements server providing the entitlements server's name (beagle.rnet.missouri.edu), the service provider username (***osprey.rnet.missouri.edu@missouri.edu***) and the service provider's virtual organization (**greatplains.net**). It also provides the administrative user username (*ciordas@missouri.edu*), the administrative user virtual network (greatplains.net) and the entitlement value that is looked up (**urn:mace:greatplains.net:repository**).

3. If the desired operation is SP_USE, the format of the client application's parameters is:

<command number (SP_USE=30)>

<Entitlements Server Machine Name>

<Service provider username@Service provider institution name>

<Service provider virtual organization name>

<Admin. User username@institution name>

<Admin. User virtual organization>

<Admin. User entitlement>

<username@institution name>

<virtual organization>

<entitlement>

<command(1 – ADD, 2 – DELETE, 3 – LOOKUP, 4 – Display entitlements, 5 – Display users, 6 – Display all, 7 - Logout)>

For instance, the following line

"*30* beagle.rnet.missouri.edu

osprey.rnet.missouri.edu@missouri.edu **greatplains.net**

ciordas@missouri.edu greatplains.net **admin**

smore@mizzou.edu greatplains.net urn:mace:greatplains.net:biosci 1"

issues a SP_USE (*30*) command to the entitlements server providing the entitlements server's name (beagle.rnet.missouri.edu), the service provider username (***osprey.rnet.missouri.edu@missouri.edu***) and the service provider's virtual organization (**greatplains.net**). It also provides the administrative user username (*ciordas@missouri.edu*), the administrative user virtual network (greatplains.net) and its administrative privilege (**admin**). The administrative user's command parameters are provided: the username (smore@mizzou.edu), the user virtual organization (greatplains.net), the entitlement value (urn:mace:greatplains.net:biosci) and the command code (1 for add a new record).

Operation:

In order to be executed properly, the client application needs to have access to: the db_client executable, the service provider private key file, the entitlements server public key file and the configuration file "key_file.txt". All these files need to be accessible to

the client application executable file. The client application generates a file called "symmetric.key" that stores the symmetric key received from the entitlements server.

The configuration file "key_file.txt" has the following format:

keys_directory = \home\keys\

client_private_key = osprey_client.private

client_public_key = osprey_client.public

server_port = 9544

server_public_key = public_key.cert

where the name on the left side of the equal sign represents the key words that describe the values on the right hand side of the equal sign. "client_private_key" is used to store the name of the client's private key file name, "client_public_key" is used to store the client's public key file name, "server_port" is used to store the entitlements server's port number, "server_public_key" is used to store the server's public key file name, and "keys_directory" is used to store the keys directory's name.

The client application starts by reading the "key_file.txt" file in order to know the names of all the needed files and the port number. This information is used throughout the client's execution.

Return values:

Upon completing its execution, the client application returns one or more of the following return values according to the operation type (SP_SETUP, SP_LOOKUP or SP_USE).

1. SP_SETUP:

SP_SETUP_SUCCESS: This value is returned in case the entitlements server succeeded in authorizing the service provider which issued the request, the server created and saved a symmetric key and a time stamp for the service provider and the client application successfully received and stored the symmetric key.

SP_AUTHENTICATION_FAILED: This value is returned if the entitlements server did not authorize the service provider to use its services.

2. SP_LOOKUP:

SP_SESSION_OK: This value is returned in case the entitlements server has verified and approved the validity of the service provider's time session.

SP_SESSION_EXPIRED: This value is sent in case the entitlements server decided that the service provider's session has expired. The entitlements server deleted the service

provider's records in the time and symkey databases that hold the sessions time stamps and the symmetric keys respectively.

USER_ENTITLEMENT_LOOKUP_FAILED: This value is returned if the entitlements server failed to find the requested combination of user credentials and entitlement value in its entitlement database.

USER_ENTITLEMENT_LOOKUP_SUCCEEDED: This value is returned if the entitlements server found the requested combination of user credentials and entitlement value in its entitlement database.

3. SP_USE:

SP_SESSION_OK: This value is returned in case the entitlements server has verified and approved the validity of the service provider's time session.

SP_SESSION_EXPIRED: This value is sent in case the entitlements server decided that the service provider's session has expired. The entitlements server deleted the service provider's records in the time and symkey databases that hold the sessions time stamps and the symmetric keys respectively.

USER_SESSION_OK: This value is returned in case the entitlements server has verified and approved the validity of the administrative user's time session.

USER_SESSION_EXPIRED: This value is sent in case the entitlements server decided that the administrative user's session has expired. The entitlements server deleted the administrative user's record in the time database that holds the sessions time stamps.

USER_AUTH_MISSING: This value is returned in case the entitlements server does not receive an administrative user entitlement that matches one of the "admin" or "root" entitlements. The server does not grant access to any user that does not present the required entitlement to use its administrative service.

ADD_ENTRY_ALREADY_EXISTS: This value is returned in case the entry that the administrative user wants to add to the entitlements database is already present.

ADD_ENTRY_SUCCESS: This value is returned if the addition of a new entry in the entitlements database succeeded.

ADD_ENTRY_FAILURE: This value is returned if the addition of a new entry in the entitlements database failed.

DELETE_ENTRY_SUCCESS: This value is returned if the deletion of an entry in the entitlements database succeeded.

DELETE_ENTRY_FAILURE: This value is returned if the deletion of an entry in the entitlements database failed.

LOOKUP_CODE_1: This value is returned in case the lookup of a user's entitlement was successful.

LOOKUP_CODE_0: This value is returned in case the lookup of a user's entitlement failed.

LOOKUP_USER_DONE: This value is returned when a display command is issued and the message is used to decide when all the records sent by the server have been received by the client application.

Error messages:

The error messages that the client application returns deal with various errors that occur such as the inability to open a file, to allocate memory, to send or receive data on the socket, to encrypt or decrypt using a public-private key approach or a symmetric approach or some of the parameters are ill-formatted.

"Please_provide_the_correct_[operation type]": This message is returned in case the operation type (SP_SETUP, SP_LOOKUP or SP_USE) are not correctly provided - SP_SETUP = 10, SP_LOOKUP = 20 and SP_USE = 30.

"Please_provide_the_ [Service DNS name]": This message is returned in case the entitlement's server name is null.

"Please_provide_the_correct_[SP_username]": This message is returned in case the service provider username is null or it doesn't comply with the issuing organization's naming convention.

"Please_provide_the_correct_[SP_institution]": This message is retuned if the service provider institution's name is null or it doesn't comply with the issuing organization's naming convention.

"Please_provide_the_correct_[SP_vo]": This message is retuned if the service provider virtual organization's name is null or it doesn't comply with the used naming convention.

"Error_opening__\"key_file.txt\"_file": This message is returned if an error occurs while trying to open the "key_file.txt" configuration file.

"Please_provide_the_correctly_formated_\"key_file.txt\": This message is returned if the "key_file.txt" is not formatted according to the format described above.

"The_local_private_key_file_is_not_accessible": This message is returned if the client's private key file is not present in the current directory.

"Error_retrieving_server_port": This message is returned in case the port number value retrieved from the “key_file.txt” configuration file is not a number.

"The_server_public_key_file_is_not_accessible": This message is returned in case the server’s public key file is not accessible from the current directory.

"db_server_connect: Socket_init_error": This message is returned in case there is an error creating a socket and connecting to the server.

"db_server_authentication: Error_allocating_memory"” This message is returned in case there is an error allocating memory for the authentication message the client is sending to the server at the beginning of any communication.

"db_server_authentication: Error_creating_encrypted_message": This message is returned in case there is an error encrypting the authentication message.

"db_server_authentication: Socket_write_error": This message is returned in case there is an error sending the messages to the server.

"SP_SETUP: Connection_closed_by_Entitlement_Server": This message is returned in case the communication channel with the server is terminated prematurely.

"SP_SETUP: Error_decrypt_Entitlement_Server_message": This message is returned in case there is an error decrypting the server's message. The error can lie in opening and reading the corresponding keys, or the read keys are not the right ones to perform the decryption.

"SP_SETUP: Error_creating_symmetric.key_file": This message is returned in case there is an error creating the symmetric.key file to store the server's symmetric key.

"SPLOOKUP_ERROR_MISSING_ARGUMENTS": This message is returned in case the number of needed arguments to perform a SP_LOOKUP operation is not correct.

"SP_LOOKUP: Please_provide_the_correct_[USER_username]": This message is returned in case the looked up username is null or it doesn't comply with the issuing organization's naming convention.

"SP_LOOKUP: Please_provide_the_correct_[USER_institution]": This message is retuned if the looked up user's institution is null or it doesn't comply with the issuing organization's naming convention.

"SP_LOOKUP: Please_provide_the_correct_[USER_vo]": This message is retuned if the looked up user's virtual organization name is null or it doesn't comply with the used naming convention.

"SP_LOOKUP: Error_open_\"symmetric.key\"_file": This message is used in case there is an error opening the symmetric key file in order to retrieve the secure communication channel symmetric key.

"SP_LOOKUP: Entitlement_Server_terminated_prematurely": This message is returned if the communication with the server is terminated unexpectedly.

"SP_USE_ERROR_MISSING_ARGUMENTS": This message is returned in case the number of needed arguments to perform a SP_USE operation is not correct.

"SP_USE: Please_provide_the_correct_[USER_username]": This message is returned in case the looked up username is null or it doesn't comply with the issuing organization's naming convention.

"SP_USE: Please_provide_the_correct_[USER_institution]": This message is retuned if the looked up user's institution name is null or it doesn't comply with the issuing organization's naming convention.

"SP_USE: Please_provide_the_correct_[USER_vo]": This message is retuned if the looked up user's virtual organization name is null or it doesn't comply with the used naming convention.

"SP_USE: Please_provide_the_correct_[username]>": This message is returned in case the username is null or it doesn't comply with the issuing organization's naming convention.

"SP_USE: Please_provide_the_correct_[institution]": This message is retuned if the institution name is null or it doesn't comply with the issuing organization's naming convention.

"SP_USE: Please_provide_the_correct_[vo]": This message is retuned if the virtual organization name is null or it doesn't comply with the used naming convention.

"operationCode_NOT_VALID": This message is returned in case the chosen command number is different from Add – 1, Delete - 2, Lookup an entitlement – 3, Display entitlements with the same user - 4, Display users with the same entitlement – 5, Display all records – 6, Logout administrative user – 7.

"SP_USE: Error_opening_\"symmetric.key\"_file": This message is used in case there is an error opening the symmetric key file in order to retrieve the secure communication channel symmetric key.

"SP_USE: Entitlement_Server_terminated_prematurely": This message is returned if the communication with the server is terminated unexpectedly.

Appendix E –Web Interface Description

A web interface to the client application is necessary due to the web-related implementation of the Shibboleth environment and also due to convenience incurred by using the web as an interface to the shared computing resources as long as they offer a web interface too. The web interface is built on top of the client application and allows for service providers and administrative users to send queries and updates to the entitlements server.

Our web interface for administrative users is comprised of one HTML page and two PHP scripts. PHP is a server-side HTML embedded scripting language. It provides web developers with a full suite of tools for building dynamic websites. PHP is chosen because of its ease in deployment and wide-availability. The web interface files are form.html, process.php and admin.php.

Form.html is the initial page that an administrative user is directed when trying to access the administrative capabilities of the entitlements server. If the user is authorized to use the entitlements server, they are displayed a page generated by the process.php PHP script that allows for various entitlement management operations to be performed. After choosing an operation and filling the required fields, the admin.php script verifies the supplied values, sends the corresponding query to the entitlements server and displays the results.

Using the administrative capabilities of the Entitlements Server

When an administrative user requests access to manage the entitlements server, he or she needs to be firstly authenticated by their home institution's identity provider. Figure 14 shows the WAYF page that a user is redirected to after typing the URL of the entitlements administrative page (e.g., https://osprey.rnet.missouri.edu/authtest/PHP/form.html) in their browser.

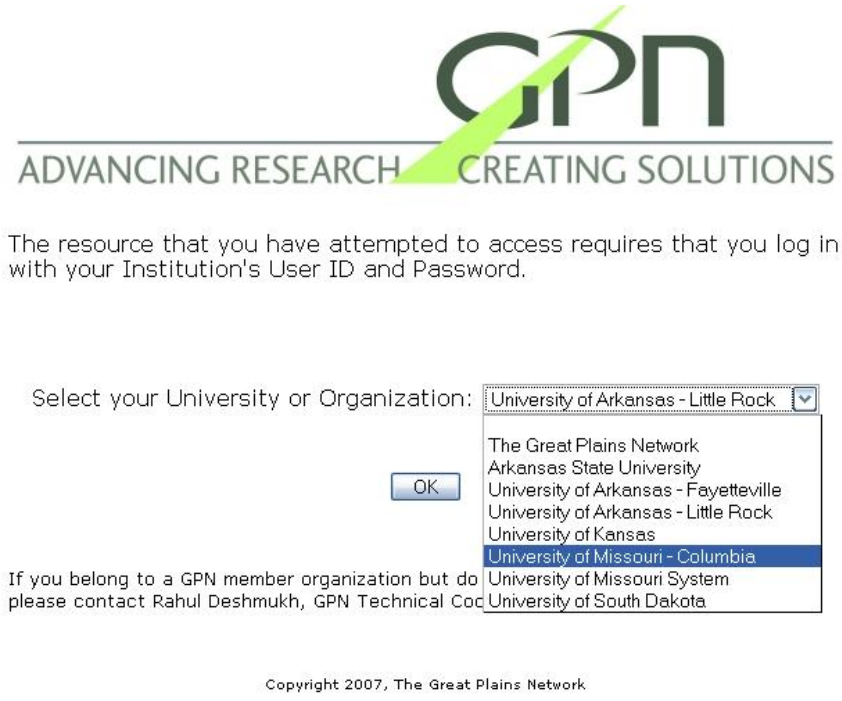

Figure 14. The Great Plains Network WAYF service

The user is prompted to choose their home institution's name and be authenticated by their home institution's identity provider. Figure 15 shows the identity provider from the University of Missouri – Columbia authenticating the administrative user by prompting them for a username and a password. Upon successful authentication by the identity provider, the administrative user is redirected to the requested page, in this case form.html.

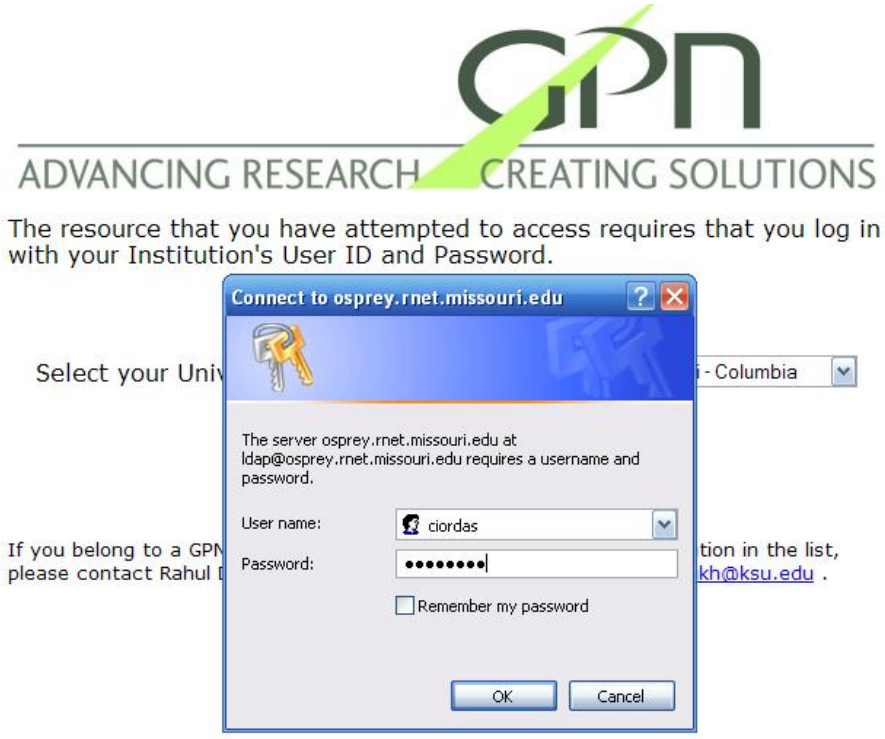

Figure 15. The identity provider from University of Missouri – Columbia authentication

Upon redirection, the administrative user has the chance to interact with the authorization page of the entitlements server. Form.html is a web page that requires the user to choose between an administrative role and a root role, and to fill out a field with the name of the

virtual organization that they request authorization to manage its entitlements. Figure 16 shows the page with the fields filled up.

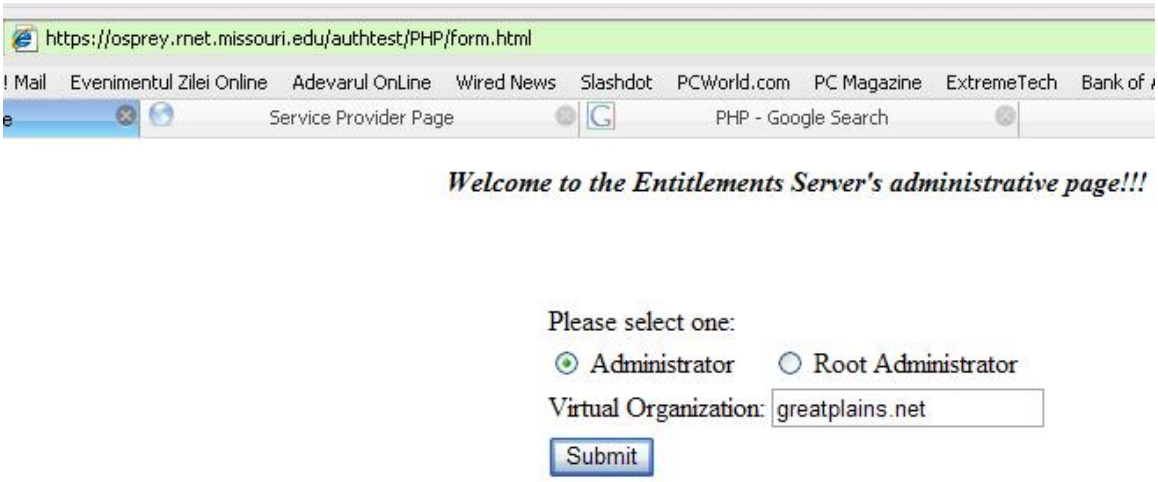

Figure 16. The Entitlements Server administrative page

Upon submitting the form, the entitlements server is contacted by the client application issuing an SP_LOOKUP operation. The operation requests the server to authorize the user that has just been authenticated by Shibboleth and who requires access to a specific virtual organization with a specified privilege role (admin or root). The server returns the authorization answer. Figure 17 depicts a successful authorization step which leads to the display of the administrative user's commands menu. Figure 18 shows a failed authorization decision that does not allow the user to manage the entitlements server.

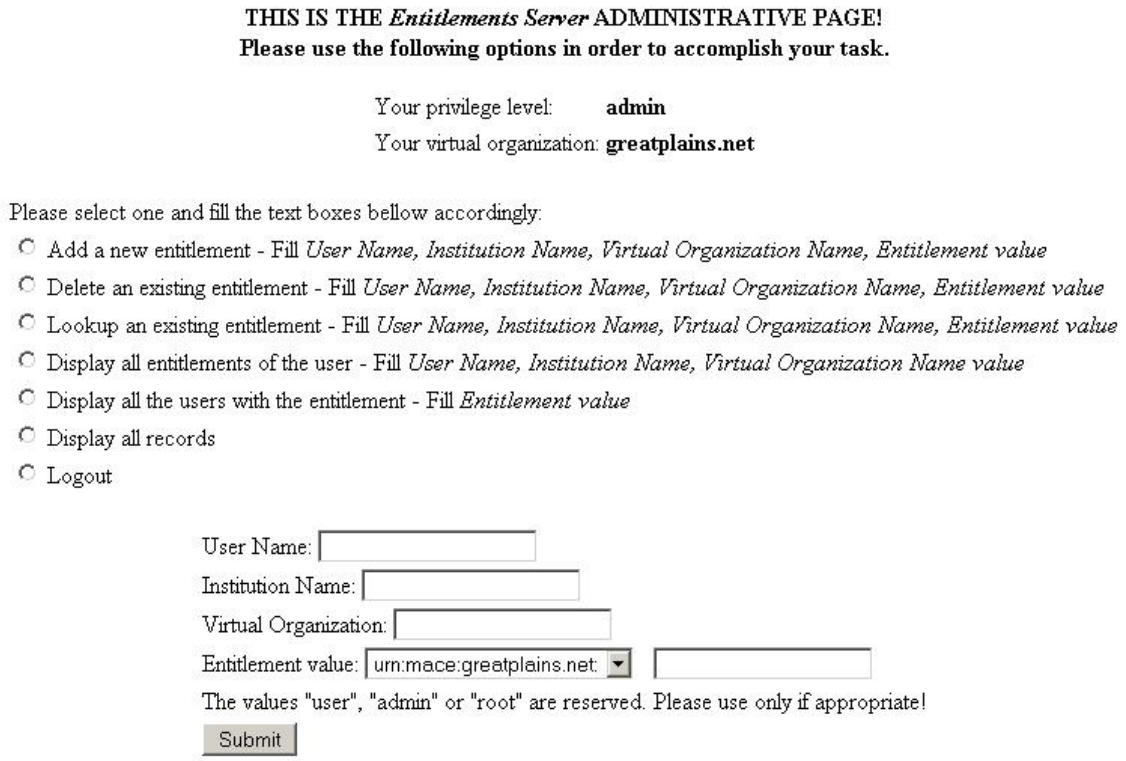

Figure 17. The administrative user is authorized to manage the entitlements server

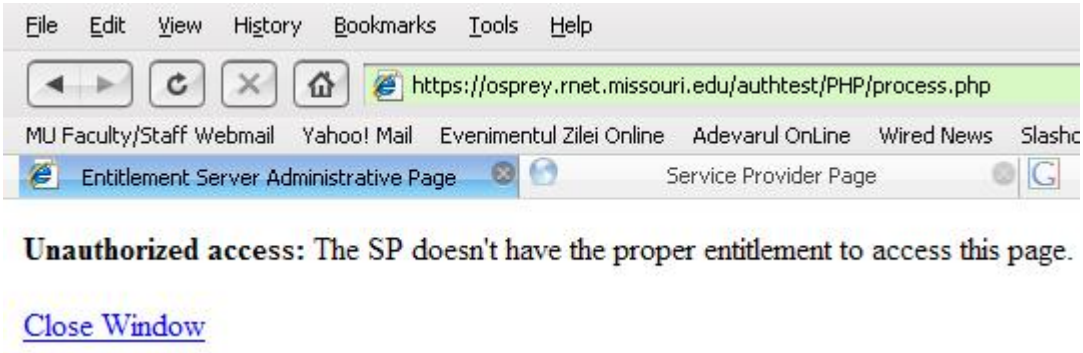

Figure 18. The administrative user failed to be authorized by the entitlements server

If the administrative user is authorized by the entitlements server to issue requests and updates, the user needs to choose one of the displayed options and fill the corresponding text boxes.

To add a new entitlement, the administrative user needs to fill the User Name, the Institution Name, the Virtual Organization and Entitlement value fields. If the administrative user has only an "admin" privilege role, he or she can only add a user with the same institution name and virtual organization such as the administrative user. This is used in order to make sure that an administrative user has can only modify users entries belonging to its own institution. Figure 19 shows the request to add a new record for a user that has Mizzou.edu as the home institution name. Because the institution name is different than the administrative user's institution name "missouri.edu", the request is denied as shown in Figure 20.

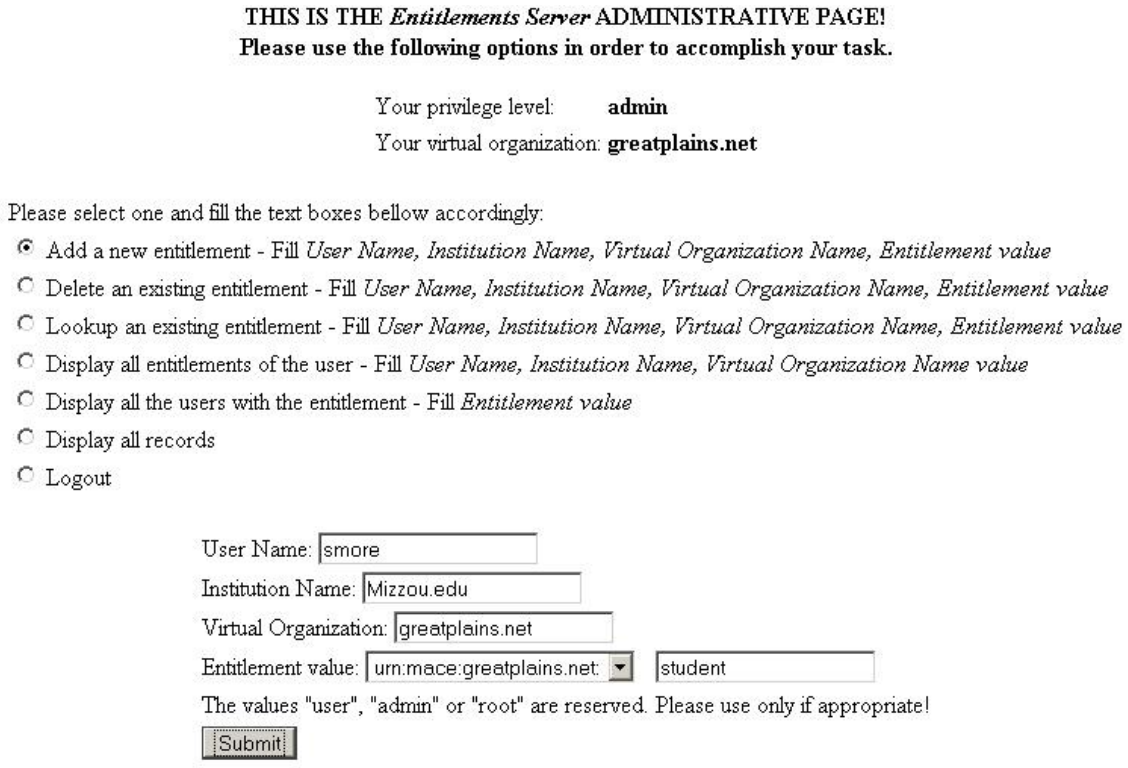

Figure 19. Add command

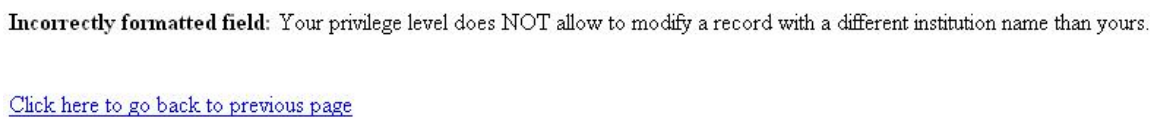

Figure 20. Invalid add request.

To delete an entitlement, the administrative user needs to fill the User Name, the Institution Name, the Virtual Organization and Entitlement value fields. Upon completion of the command, the client application will return a message: DELETE_ENTRY_SUCCESS or DELETE_ENTRY_FAILURE.

To lookup an existing entitlement, the administrative user needs to fill the User Name, the Institution Name, and the Virtual Organization and Entitlement value fields. Once completed, the client application sends a message of LOOKUP_CODE_1 in case of a successful search or LOOKUP_CODE_0 otherwise.

To display all existing entitlements that have a specified user credentials, the administrative user needs to fill the User Name, the Institution Name, and the Virtual Organization. The client application will return a list of values. Figure 21 shows the chosen options, while Figure 22 displays the results of the query.

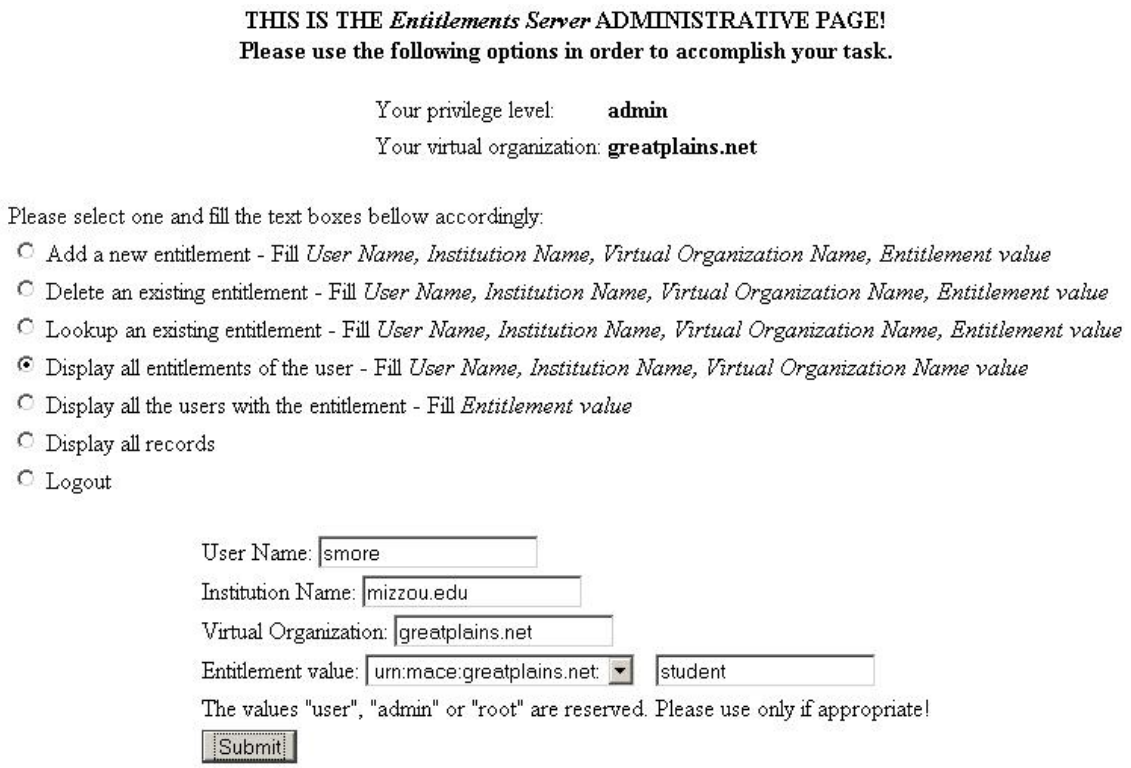

Figure 21. Display all the entitlements of the current user.

Index	VO	Institution	User Name	Entitlement
1	greatplains.net	mizzou.edu	smore	urn:mace:greatplains.net:biogrid
2	greatplains.net	mizzou.edu	smore	user
3	greatplains.net	mizzou.edu	smore	urn:mace:greatplains.net:uark.edu:webmpi

Click to the Administrative Page

Figure 22. The results of the "Display all the entitlements of the current user" command

To display all existing users that have a specified entitlement, the administrative user needs to fill the Entitlement value. The client application will return a list of values.

To display all the records that the administrative user's privilege level allows, the user does not need to fill any of the provided fields. The client application will return a list of records provided by the entitlements server.

To logout the current administrative user from the current session, the Logout option needs to be selected and submitted. The entitlements server will delete the administrative user's time record in the time database and the administrative user will be redirected to the original authorization page, form.html.

Glossary

Attribute -- represents a specific characteristic of a person or object such as the home institution's name, the university status or what entitlements a user might have for certain services.

Authentication -- is the process of establishing if a user is the person or program that they claim to be [AUTHN].

Authorization -- is the process of determining, after a user is authenticated, if a user is allowed to have access to a particular resource [AUTHZ].

Certificate Authorities (CAs) -- is an entity which issues public-key certificates for use by other parties.

Entitlement -- is a guarantee of access to benefits because of rights, or by agreement through law or agreed upon policies.

eduPerson Object Class -- provides a common list of attributes and definitions for use in the higher education identity management systems to store user's identity [HAZELTON].

Federation – Associations of enterprises that come together to exchange information about their users and resources in order to enable collaborations and transactions [GETTES]. A federation provides part of the underlying trust required for function of the Shibboleth architecture. A federation is a group of organizations (universities,

corporations, content providers, etc.) who agree to exchange attributes using the SAML/Shibboleth protocols and abide by a common set of policies and practices.

Great Plains Network (GPN) -- is a consortium of universities in the Midwestern states of Arkansas, Kansas, Missouri, Nebraska, North Dakota, Oklahoma, South Dakota dedicated to supporting research and education through the use of advanced networking technology [GPN].

Identity management system -- is a application or system that deals with the management of the identity life cycle of entities (subjects or objects) during which the identity is established - a name (or number) is connected to the subject or object; the identity is described - one or more attributes which are applicable to the particular subject or object may be assigned to the identity or changed as needed; and the identity is destroyed [IDM].

Internet2 -- the most advanced networking consortium in the United States. It has been formed and is led by the research and education community. Internet2 promotes the educational and creative missions of its members by providing both leading-edge network capabilities and unique partnership opportunities to facilitate the development, deployment and use of revolutionary internet technologies [INT2].

LDAP (Lightweight Directory Access Protocol) - a protocol used to access and modify information stored in an information directory.

MACE (Middleware Architecture Committee for Education) - a group of leading higher education IT architects that provide overall direction and vision for the Internet2 Middleware Initiative. The goal of the Internet2 Middleware Initiative is to contribute to the building of an international interoperable middleware infrastructure for research and education [MACE].

Middleware - is a general term for any software packages that serve to "glue together" or mediate between two separate and often already existing software systems. In the GPN case, in order for users to use shared resources seamlessly, they have created Shibboleth to be a middle layer of software that quietly deals with all the communication needed to authenticate and authorize a user.

OASIS (the Organization for the Advancement of Structured Information Standards) -- is a global consortium that drives the development, convergence and adoption of e-business and web service standards.

Privacy - the ability to control what information one reveals about oneself over the Internet, and to control who can access that information.

Public key certificate -- is a certificate which uses a digital signature to bind together a public key with an identity - information such as the name of a person or an organization and their address. The certificate can be used to verify that a public key belongs to an individual [PKCERT].

Public Key Infrastructure (PKI) - is an arrangement that provides for binding of public keys to users [PKCERT].

SAML (Security Assertion Markup Language) – is an XML standard for exchanging authentication and authorization data between security domains - between an identity provider (a producer of assertions) and a service provider (a consumer of assertions).

Shibboleth service provider – represents the entity that communicates with the user, the user's identity management system at her home institution and makes the access control decision based on the user's entitlements [SHIBINTRO].

Shibboleth identity provider -- represents the entity that authenticates a user and answers attributes inquiries from the service provider. The identity provider is embedded in the university's identity management system [SHIBINTRO].

Virtual organization – comprises a set of legally independent organizations that share resources and skills to achieve its mission, but that are not limited to an alliance of for profit enterprises. The interaction among members of the virtual organization is mainly done through computer networks. Virtual organizations are a manifestation of academic collaboration.

Windows Active Directory - is an implementation of the LDAP protocol by Microsoft. Active Directory allows administrators to assign enterprise-wide policies, deploy programs to many computers, and apply critical updates to an entire organization. An Active Directory stores information and settings relating to an organization in a central, organized, accessible database.

XML (eXtended Markup Language) - is a W3C-recommended general-purpose markup language that supports a wide variety of applications. Its primary purpose is to facilitate the sharing of data across different information systems, particularly systems connected via the Internet [XML].