INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Management von Mitgliedschaften zu Virtuellen Organisationen im Grid

Marcin Czyzewski

Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering
Betreuer: Michael Schiffers

Abgabetermin: 11. April 2006

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Management von Mitgliedschaften zu Virtuellen Organisationen im Grid

Marcin Czyzewski

Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering
Betreuer: Michael Schiffers

Abgabetermin: 11. April 2006

Hiermit versichere ich, dass ich das vorliegende Fortgeschrittenenpraktikum selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 11. April 2006

. . . . . .

(Unterschrift des Kandidaten)

Zusammenfassung

Die stetig wachsenden Anforderungen an die Rechenleistung und -kapazität werden zunehmend nur noch durch eine Grid-Infrastruktur erfüllt werden können. Der Zugriff auf die bereitgestellten Dienste, Ressourcen, Software und Daten jeglicher Art wird über das Konzept von Virtuellen Organisationen (VO) gestattet. Anwender müssen zu Mitgliedern in einer VO werden, um im Grid arbeiten zu können. Für die Verwaltung der Mitglieder wird eine Management-Software benötigt. Im Rahmen dieses Fortgeschrittenenpraktikums wird dafür die Software VOMS verwendet. Diese wird auf der Grid-Infrastruktur des Lehrstuhls installiert. Als Middleware wird Globus Toolkit vorausgesetzt. Nach erfolgreicher Installation können Mitgliedern einer VO Rechte zugewiesen werden und Mitglieder in Gruppen zusammengefasst werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Der Intel-Mitbegründer Gordon Moore schrieb 1965, dass die Transistoranzahl auf einer integrierten Schaltung mit der Zeit exponentiell ansteigt. Damit einhergehend steigt natürlich auch die Rechenleistung. Trotz dieser enormen Entwicklung ergeben sich in der Praxis immer wieder Probleme, die selbst mit der Rechenkraft eines Rechenzentrums nicht in gewünschter Zeit gelöst werden können. Ein Beispiel dafür ist die Auswertung von riesigen Datenmengen. Einer der größten Datenerzeuger der Welt wird nach der Fertigstellung im Jahre 2007 der Teilchenbeschleuniger des **European Organization for Nuclear Research** (**CERN**) [cer] in Genf sein. Pro Jahr werden dann durch die durchgeführten Versuche ca. 15 Petabyte an Daten entstehen, die man zwischenlagern und auswerten muss. Einzelne Rechner würden dieser Aufgabe nicht gerecht werden können, ein eigenes Rechenzentrum kommt aus Kostengründen nicht in Frage. Die Lösung ist die Aufteilung der Daten in Pakete und die Weitergabe dieser Datenpakete an einzelne Rechenzentren in der ganzen Welt. Diese werden dann zu einem Grid zusammengeschlossen und senden die Auswertungen der Daten zurück an das CERN. Hierbei erspart sich das CERN nicht nur die reine Auswertungsarbeit sondern auch das Zwischenlagern der enormen Datenmengen.

Sowohl in der Wirtschaft als auch in der Forschung wurde das enorme Potential von Grids erkannt. So engagieren sich neben namenhaften Unternehmen wie IBM oder Hewlett Packard auch Staaten und Staatengemeinschaften wie die EU, um den Aufbau von Infrastrukturen voranzutreiben.

Forschungseinrichtungen aus 27 Europäischen Staaten beteiligen sich am EGEE [EGE] Projekt. Dieses hat vorrangig drei Ziele. Neben dem Aufbau eines Grids soll mit der ständigen Betreuung und Verbesserung der Middleware ein verlässlicher Service für die Nutzer geboten werden. Neben dieser kontinuierlichen Optimierung wird die Zielgruppe auch immer weiter vergrößert, um den maximalen Nutzen für alle Beteiligten zu generieren. Somit sollen auch immer weitere Nutzer aus Forschung und Wirtschaft gewonnen werden. Beim Aufbau der Infrastruktur profitiert man von den bisherigen Projekten, die von den Teilnehmern bisher umgesetzt wurden. Es ist eines der größten Projekte dieser Art und wird von der EU mit 30 Millionen Euro subventioniert.

Auch auf nationaler Ebene werden Infrastrukturen aufgebaut. Bereits 2003 wurde von der Wissenschaft ein erstes Strategiepapier im Rahmen der D-Grid Initiative [d-g] veröffentlicht. Ziel ist es, eine Grid-Infrastruktur aufzubauen, die der gesamten deutschen Wissenschaft zur Verfügung stehen soll. Damit wird den Wissenschaftlern ermöglicht, auf gemeinsame Ressourcen zuzugreifen und Forschungsergebnisse untereinander einfacher auszutauschen. Insgesamt wird eine effizientere Zusammenarbeit ermöglicht. Die D-Grid Initiative wurde dann 2004 vom BMBF in die deutsche eScience Initiative eingegliedert.

Ein wichtiges Konzept, das bei Grids verwendet wird, ist das der **Virtuellen Organisationen** (**VO**). Es ist eine Form der Organisation von Einzelpersonen oder Unternehmen zu einem virtuellen Verbund um ein gemeinsames Ziel zu erreichen. Im Grid werden Virtuelle Organisationen dazu genutzt, den Ressourcenzugriff zu beschränken. Somit wird bestimmten Benutzergruppen nur die Nutzung bestimmter Ressourcen ermöglicht. Zur Umsetzung wird eine Managementsoftware benötigt. Ein Beispiel hierfür ist das im Rah-

men dieser Projektarbeit eingesetzte **Virtual Organization Membership Service** (**VOMS**) [vom c], welches auf einer Datenbank basiert. Die Installation setzt ein Grid voraus, das mit dem Open Source Produkt „Globus Toolkit“ [gt] eingerichtet wurde. Dieses wird von der Globus Alliance [ga] entwickelt. Ist so ein Grid vorhanden, kann VOMS nachträglich installiert und dann das gesamte Mitgliedsmanagement darüber realisiert werden.

Kapitel 2

Virtuelle Organisation und VOMS

Im Folgenden wird auf die Arbeitsweise rund um eine Virtuelle Organisation (VO) eingegangen und welche Anforderungen sich daraus für ein VO-Management-System ergeben. Daraufhin wird die dafür entwickelte Softwarelösung VOMS (Virtual Organization Membership Service) [vom c] vorgestellt.

2.1 Anforderungen an ein VO-Management-System

Ein Management-System ist immer nur ein Teil eines größeren Systems und sollte sich somit möglichst nahtlos und problemlos in Systeme einfügen lassen, die nach gängigen Architekturen aufgebaut sind bzw. die auf marktbeherreschenden Softwarelösungen basieren. Damit ist das Zurückgreifen auf vereinbarte Standards unumgänglich. Beispiel für so einen Standard ist die **Extensible Markup Language** (**XML**). Viele Programme bieten die Möglichkeit, Steuerinformationen oder Anfragen zu stellen, die unter Einhaltung der Spezifikation erstellt wurden. Auf der anderen Seite ist es einer Management-Software möglich, Schnittstellen zur Interaktion mit anderen Software-Modulen bereitzustellen, indem sie in XML formulierte Anfragen verarbeitet. In diesem Zusammenhang bietet sich die **Security Assertion Markup Language** (**SAML**) an. Diese XML-basierte Sprache stellt Funktionen bereit, um sicherheitsbezogene Informationen zu beschreiben und zu übertragen. Ein weiterer Standard, der sich in diesem Feld anbietet, ist das X.509 Zertfikat. Es kann zur sicheren Authentifikation verwendet werden. Dadurch kann sichergestellt werden, dass nur berechtigte Personen Zugriff auf Ressourcen erhalten. Außerdem kann man damit bei missbräuchlicher Nutzung von Ressourcen den entsprechende Benutzer identifizieren. In diesem Zusammenhang ist das Logging ein wichtiger Aspekt, der bei der Konzeptionierung nicht vernachlässigt werden sollte. Nur dadurch wird es möglich, auch im Nachhinein bestimmte Ressourcenzugriffe einzelnen Benutzern zuzuordnen.

Beim Entwickeln eines VO-Management-Systems sollte darauf geachtet werden, dass es nicht ausschließlich auf den Einsatz in Kombination mit einer einzelnen Grid-Middleware ausgelegt ist. Auch wenn sich das Globus Toolkit als Quasi-Standard etabliert hat, so werden auch andere Softwarelösungen eingesetzt und die weitere Entwicklung kann nicht abgesehen werden. Somit sollten mindestens Portierungsmöglichkeiten auf andere Middleware-Lösungen ermöglicht werden.

Die Funktionsweise aus Benutzersicht sollte möglichst einfach gestaltet sein. Wünschenswert ist hier eine einmalige Authentifikation beim Beginn des Arbeitsvorgangs und jegliche Authorisation sollte dann über ein bestehendes Zertifikat im Hintergrund abgewickelt werden.

Auch die Administration sollte durch möglichst einfache Interaktion möglich sein. Der Administrator wird diese Aufgabe in den meisten Fällen nur nebenbei erledigen wollen, so dass eine lange Einarbeitung oder komplizierte Administrierungsvorgänge hier hinderlich wären.

Ein weiterer wichtiger Aspekt eines VO-Management-Systems ist es, den Benutzern zu ermöglichen, Mitglied in mehreren Virtuellen Organisationen zu werden. Diese Zugehörigkeit zu verschiedenen VOs ist zwingend erforderlich, da in der Praxis, insbesondere für wissenschaftliche Mitarbeiter, das Mitwirken an mehreren Projekten möglich sein muss. Weiterhin sollte es innerhalb einer VO möglich sein, eine gewisse Hierarchie aufzubauen. Dies kann z.B. durch die Zuweisung von Rollen realisiert werden, denkbar sind hier Rollen wie Administrator, Hauptbenutzer oder Nutzer mit eingeschränkten Rechten.

2.2 VOMS

Im Rahmen dieses Fortgeschrittenenpraktikums wurde als VO-Management-Software der Virtual Organization Membership Service (VOMS) ausgewählt. Diese wurde im Rahmen des DataGrid Projekts entwickelt, welches von der EU finanziert wurde. Die Ergebnisse aus diesem Projekt werden momentan im Rahmen des Projektes **Enabling Grids for E-SciencE** (**EGEE**) genutzt und weiterentwickelt.

Bei VOMS wird zur Authentifikation ein X.509 Zertifikat vorausgesetzt, welches von einer **Certificate-Authority** (**CA**) ausgestellt werden muss. Die Zertifizierungsstelle stellt die Identität eines Antragstellers fest und beglaubigt das Zertifikat durch das Signieren mit ihrer eigenen digitalen Signatur. Für Deutschland ist hierbei die DFN-Cert Services GmbH der Ansprechpartner. Um diese oberste Zertifizierungsstelle zu entlasten und es Benutzern zu ermöglichen sich auch an anderen Orten einer Identitätsprüfung zu unterziehen gibt es spezielle Institutionen die ebenfalls dazu berechtigt sind, Anträge für Zertifikate engegenzunehmen. Dazu zählt auch die im vorliegenden Fall benutzte Certificate-Authority, nämlich das Leibniz-Rechenzentrum München (LRZ).

Die lokale Authorisation im Grid erfolgt auf der Grundlage von temporär ausgestellten Zertifikaten. Um so ein Zertfikat zu erhalten, muss der Benutzer sich anhand seines durch eine CA signierten Zertifikats beim VOMS-Server authorisieren. Dabei kann er die Rechte anfordern, die ihm aufgrund seiner Zugehörigkeit zu einer Virtuellen Organisation zustehen. Nach einer Überprüfung in der Datenbank erstellt der VOMS-Server daraufhin ein Zertifikat, welches dann diese Berechtigungen des Benutzers innerhalb des Grids enthält.

Der für eine Virtuelle Organisation eingerichtete VOMS-Server enthält immer alle Benutzerkonten und die entsprechenden Rechte aller Benutzer. Die temporär ausgestellten Zertifikate machen es möglich, dass bei jeder weiteren Ressourcenanfrage kein Netzwerkverkehr entsteht. Die zugehörige Erlaubnis für den Zugriff auf eine Ressource ist ja bereits im temporären Zertifikat enthalten.

Server und Client sind in C++ implementiert, die Komponenten zur Administrierung wurden jedoch in Java geschrieben. Zur Registrierung von neuen Virtuellen Organisationen sowie zur Administrierung kann ein Web Interface genutzt werden.

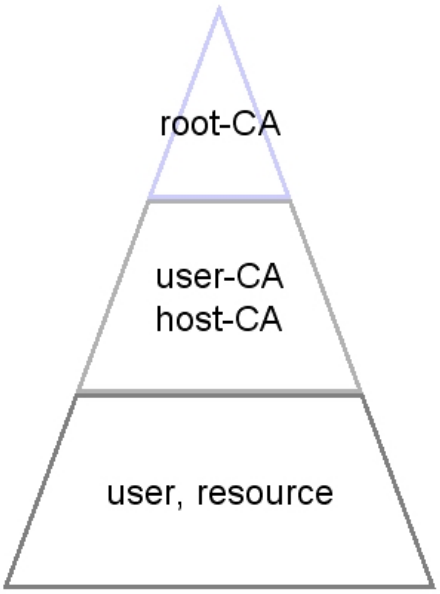

Abbildung 2.1: Hierarchie bei Zertifikaten, aus [fop]

VOMS benötigt eine relationale Datenbank, um die Benutzerrechte ablegen und verwalten zu können. Darin werden neben dem aktuellen Zustand auch vergangene Zustände abgelegt, so dass ein Nachvollziehen vergangener Zustände ermöglicht wird. In der aktuellen Version kann als Datenbank sowohl MySQL als auch Oracle verwendet werden.
Weiterhin wird ein Apache Webserver benötigt. Durch diesen wird ein Webinterface zur Verfügung gestellt. Dieses bietet dem Administrator einer Virtuellen Organisation ein bequemes Frontend über das er jegliche Verwaltungsaufgaben durchführen kann, wie z.B. Benutzer zu Gruppen hinzufügen oder neue Gruppen

anlegen.
Es ist sowohl in Grids mit Globus 2 als auch mit Globus 4 einsetzbar.

VOMS wird als Open-Source-Projekt entwickelt und das Einsatzgebiet ist eher in der Wissenschaft zu sehen, insbesondere in e-Science-Projekten.

Die Architektur von VOMS wird in Kapitel 4 näher vorgestellt, während in Kapitel 5 auf die Installation und Benutzung von VOMS eingegangen wird.

Kapitel 3

State of the Art

Im Folgenden werden weitere Management-Systeme für Virtuelle Organisationen vorgestellt.

3.1 Liberty

Liberty [lib] ist lediglich eine Spezifikation, die von der Liberty Alliance entwickelt wurde. Somit gibt es unterschiedliche Management-Software, die diese umsetzt. Bei allen Produkten werden zwar die Basiskonzepte realisiert, die die Liberty Spezifikation vorschreibt, jedoch sind diese Produkte natürlich nicht in allen Punkten identisch, da die verschiedenen Hersteller unterschiedlich an die Lösung der gestellten Probleme herangegangen sind. Ein Vertreter, der die Liberty Spezifikation umsetzt, ist der Sun Access Manager [sun].
Liberty setzt voraus, dass jeder Benutzer bei den einzelnen Partnern ein Benutzerkonto hat. Den einzelnen Partnern bleibt die Gestaltung des Logins selbst überlassen. Gegenüber weiteren Partnern kann die Authentifikation dann über Simplified Sign On erfolgen. Authorisierungsinformationen können unter den Partnern über Web Services Framework (WSF) ausgetauscht werden. Dies schreibt aber nicht direkt vor, welche Zugriffsmöglichkeiten der Partner, der die Informationen erhält, dem Benutzer gestatten wird. Darüber entscheidet dann der lokale Access-Manager.
Vorgaben für die Administration oder für das Login-Interface enthält die Spezifikation nicht, so dass die Lösung dieser Fragestellungen der Implementierung des jeweiligen Produktes überlassen bleibt. Die Sprache, in der die Spezifikation umgesetzt wird, ist ebenfalls freigestellt.
Der Sun Access-Manager, welcher die Liberty Spezifikationen erfüllt, setzt einen Webcontainer sowie einen Sun Directory Server für die Policies und LDAPv3 Directory für User voraus.
Die Spezifikation von Liberty ist frei verfügbar, die Implementierungen sind aber kostenpflichtig und so werden Liberty-Produkte eher im kommerziellen Bereich sowie von staatlichen Institutionen eingesetzt.

3.2 Shibboleth

Shibboleth [shi] wird von **Middleware Architecture Committee for Educationist** (**MACE**) entwickelt. Es ist eine Open-Source Software, bei deren Entwicklung darauf geachtet wurde sich möglichst an vorhandenen Standards zu orientieren. So wurde Shibboleth nach der Spezifikation OASIS SAML v1.1 implementiert. Demzufolge nutzt es z.B. OpenSAML als Nachrichtenformat. Shibboleth ermöglicht die Authentifikation anhand von **Web Single SignOn** (**Web-SSO**).
Shibboleth bietet auch Features im Hinblick auf die Privacy-Einhaltung. Dem Benutzer wird ermöglicht, die Informationen, die er an einen Service Provider zur Authorisationszwecken schickt, zu beschränken. So werden nicht alle Rechte offengelegt, sondern nur die für den aktuellen Zugriff erforderlichen versendet.

So muss die Identität des Benutzers nicht einmal zwingend übermittelt werden, je nach Service Provider sind auch schon die reinen Berechtigungsinformationen ausreichend.
Haupteinsatzgebiet ist, den Zugriff auf Online Inhalte individuell je nach Zugriffsberechtigung zu beschränken. Für den Einsatz in Grids wurde GridShib [gri] entwickelt. Es integriert die Shibboleth Authorisations-Infrastruktur in eine vorhandene Grid Umgebung und ist hierbei auf das Globus Toolkit ausgelegt. Wie bei VOMS werden auch hier X.509-Zertifikate verwendet. Service Provider im Grid können die Rechte der einzelnen Benutzer vom Shibboleth Identity Provider erfragen.

Von Nachteil kann hierbei das Haupteinsatzgebiet von Shibboleth sein. Es ist nun mal von der Konzeption her auf WebServices ausgelegt. Und so können Details in der Implementierung von GridShib durchaus mit der Konzeption von Shibboleth kollidieren. Problematisch könnte hierbei außerdem sein, dass Weiterentwicklungen und Fehlerbeseitigung an Shibboleth erst verspätet Einzug in GridShib finden.

3.3 Permis

Permis [per] wurde von der **Information Systems Security Research Group** (**ISSRG**) an der University of Kent (UK) entwickelt. Bei Permis handelt es sich um eine Open-Source Authorisationssoftware, die in Java geschrieben wurde. Unter Benutzung von SOAP und SAML ist Permis kombinierbar mit dem Globus Toolkit.
Permis ist ein auf Policies basierendes Authorisations-System. Die Policies werden nach dem XML Standard erzeugt. Ein System-Administrator kann eine Policy erstellen in der näher spezifiziert wird, aus welchen Rollen sich welche Zugriffsrechte im Grid ergeben. Ein VO-Administrator kann dann seinerseits durch Erstellen von weiteren Policies angeben, welche Benutzer welche Rollen annehmen können.

Die Art des Umgangs mit Zertifikaten ist sogleich ein großer Nachteil von Permis: Zertifikate werden nicht dynamisch nach dem Eingang einer Anfrage gefolgt von einer Authorisierung erstellt, sondern liegen bereits für jeden Benutzer bezogen auf eine Virtuelle Organisation vor. Authorisiert sich ein Benutzer bei einer VO, so wird das entsprechende Zertifikat aus dem Repository geholt. Problematisch hierbei ist die in der Praxis durchaus notwendige Zugehörigkeit von Benutzern zu mehreren Virtuellen Organisationen. Ein Anwender kann hier aber nur ein Zertifikat anfordern, welches ihn als Mitglied einer VO auftreten lässt. Benötigt er den Zugriff auf Dienste im Grid, die erst durch die Zugehörigkeit zu einer weiteren VO für ihn nutzbar sind, so muss er sich erneut authorisieren, was sehr auf Dauer sehr umständlich werden kann.

3.4 CAS

Der Community Authorisation Service (CAS) [cas] wurde im Rahmen des Globus Projekts entwickelt. Die Architektur verwendet Public-Key Authentifikation und basiert auf der im Globus Toolkit enthaltenen Grid Security Infrastructure (GSI). GSI ist eine Ansammlung von Bibliotheken und Tools, die Benutzern und Diensten den gesicherten Zugriff auf Ressourcen ermöglichen. Ähnlich wie bei VOMS wird auch hier nach der Authentifikation eines Benutzers ein Proxy-Zertifikat erstellt, über das dann die dem Benutzer erlaubten Zugriffe im Grid erkannt und gewährt werden können.

Im Vergleich zu VOMS gibt es einen entscheidenden Unterschied: mit CAS Attributen können Zugriffsrechte im low-level Bereich spezifiziert werden, also z.B. die Zugriffsrechte auf eine Datei. VOMS Attribute sind eher allgemeiner gehalten und es werden eher Rollen verwendet aus denen globalere Zugriffsrechte folgen.

Die Integration von Community Authorisation Service in ein bestehendes Grid, welches Globus Toolkit als Middleware verwendet ist jedoch umständlich. Die Dienste im Grid müssen dazu erst modifiziert werden. Grund dafür ist die Art der Zertifikate, die von CAS erstellt werden. Diese entsprechen nicht dem gängigen Globus Standard, der es Diensten sofort ermöglicht, den Information auszulesen, wer der Eigentümer eines Zertifikats ist. Diese Information wird von CAS anders verpackt, so dass alle Grid Dienste erst angepasst

werden müssten, um CAS-Zertfikaten verarbeiten zu können.
Ein Weiterer Nachteil von CAS ist, dass es nur einzelne Genehmigungen vorsieht, jedoch keine Gruppen oder Rollen ermöglicht. Dies erschwert die Arbeit eines Administrators erheblich, da es erst über die Einteilung in Gruppen und Zuweisung von Rollen möglich wird, eine vernünftige Hierarchie innerhalb einer Virtuellen Organisation aufzubauen.

3.5 Gemeinsamkeiten

Die Granularität, mit der Benutzerrechte vergeben werden können, ist bei allen Management-Systemen beliebig fein möglich. Ein anonymes Nutzen des Grids wird bei Liberty und GridShib ermöglicht, bei VOMS hingegen wird jeder Nutzer durch sein Zertifikat eindeutig identifiziert. VOMS wird im Rahmen des EGEE Projektes so eingesetzt, daß pro VOMS-Server 100-1000 Mitglieder registriert sind, technisch möglich wären aber wohl einige Tausend pro Server. Liberty Implementierungen sind darauf ausgelegt, einige Zehntausend Benutzer verwalten zu können. Bei Shibboleth gibt es keine Begrenzung, was die Benutzerzahlen angeht. Eine Erweiterung auf mehr Benutzer sollte aber bei allen Management-Systemen möglich sein.

Kapitel 4

Architektur von VOMS

4.1 Integration in eine bestehende Grid-Middleware

Im ersten Schritt wird eine Grid Umgebung betrachtet, die ohne Mitgliedermanagement auskommt. Möchte ein Benutzer auf das Grid zugreifen, so muss er sich dafür mit `grid-proxy-init` ein Proxy-Zertifikat erstellen. Daraufhin kann aber jeder Benutzer den Zugriff auf Ressourcen im Grid anfordern und wird diesen auch gewährt bekommen (Verfügbarkeit wird vorausgesetzt).

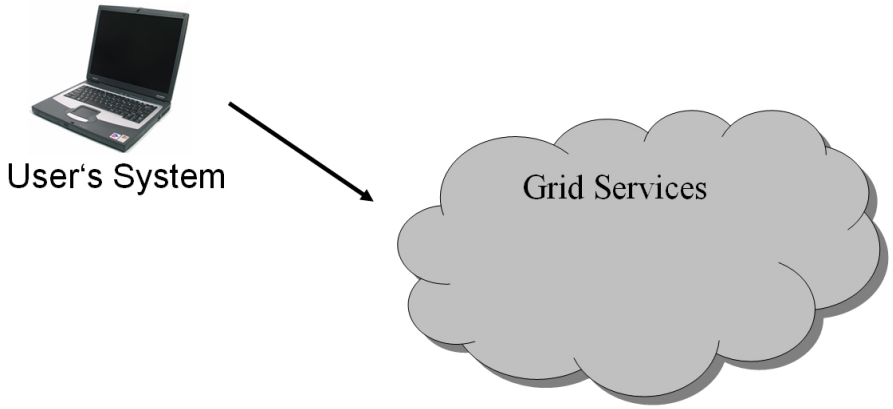

Abbildung 4.1: Grid ohne Mitgliedsmanagement

Die Mitgliedsmanagement Lösung VOMS kann in so ein bestehendes Grid nachträglich integriert werden. Die einzelnen Dienste müssen dafür „VOMS-aware“ gemacht werden. Ziel ist es, Benutzern den Zugriff auf einen Dienst nur zu ermöglichen, wenn der anfragende Benutzer die Berechtigung zur Nutzung dieses Dienstes hat.

Um das VOMS-System einzurichten, wird ein VOMS-Server benötigt, der von einem Administrator angelegt und verwaltet wird.

Möchte ein Benutzer im Grid arbeiten, so muss er sich dafür weiterhin ein Proxy-Zertifikat erstellen, nun aber statt mit `grid-proxy-init` durch den Aufruf von `voms-proxy-init`. Dieses Zertifikat enthält nun neben den reinen Identifikationsinformationen auch die einzelnen Zugriffsrechte, die der Benutzer im Grid hat. Fordert er nun eine Ressource im Grid an, so kann anhand des Zertifikats geprüft werden, ob der Benutzer diese Ressource nutzen darf.

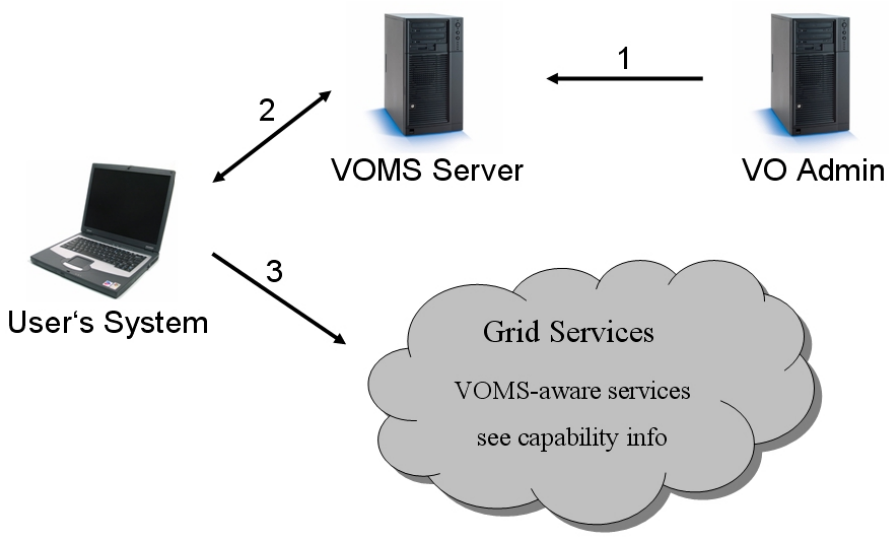

Abbildung 4.2: Grid mit Mitgliedsmanagement durch VOMS

4.2 Server und Client

VOMS kann prinzipiell in seine Server- und Client-Bestandteile unterteilt werden.

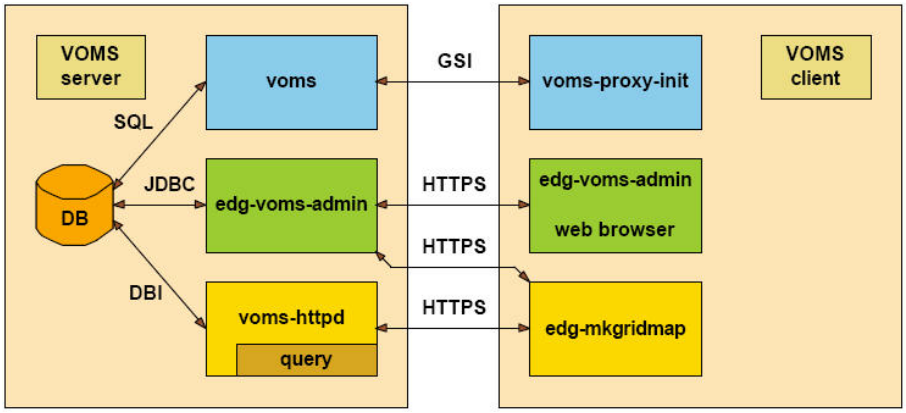

Abbildung 4.3: Die VOMS Architektur

Intern unterscheidet man aber detaillierter:

- User Server:
  erhält Anfragen von Clients und gibt Informationen über den angefragten Benutzer zurück.
- User Client
  Stellt eine Anfrage an den Server, wobei die Authorisierung anhand des Zertifikats des Benutzers vorgenommen wird. Zurückgeliefert wird eine Liste von Gruppen, Rollen und Rechten des Benutzers.
- Administration Client
  Erlaubt es einem VO Admin, jegliche Änderungen vorzunehmen wie z.B. Gruppen zu definieren, neue Benutzer anzulegen, ihre Rechte oder Gruppenzugehörigkeit zu ändern.

- Administration Server
  nimmt Anfragen von Clients entgegen und führt Änderungen an der Datenbank durch.

4.3 Datenbank

Als Datenbank kann sowohl MySQL als auch Oracle verwendet werden. In dieser werden alle Informationen zu den Mitgliedern einer Virtuellen Organisation gehalten. Damit der Administrator einer VO nicht direkt mit den Tabelleneinträgen arbeiten muss, dient der Server als Frontend für die Datenbank. Anfragen können sowohl über das Webinterface als auch über die Kommandozeile gestellt werden. In Textform sind folgende Anfragearten möglich:

A	liefert alle Informationen zum Benutzer
B<group>:<role>	liefert alle Informationen zum Benutzer, die sich aus seiner Zugehörigkeit zu der Gruppe und seinem Besitz der Rolle ergeben
G<group>	liefert alle Informationen zum Benutzer, die sich aus seiner Zugehörigkeit zur Gruppe ergeben
L	liefert eine Liste von allen möglichen Anfragen
R<role>	liefert Informationen zu der angegebenen Rolle

Alle Anfragen enthalten zusätlich implizit ein Feld userid. Dieses wird automatisch passend gefüllt, indem die Information aus dem Zertifikat entnommen wird, welches der Benutzer beim Stellen seiner Anfrage übermittelt hat.

Nachfolgend wird die Struktur der Tabelleneinträge in der Datenbank aufgelistet. Die Datentypen der Einträge sind nur sehr allgemein angegeben, da sie je nach eingesetztem Typ der Datenbank variieren können. Die Primärschlüssel sind jeweils als erstes aufgeführt und durch einen doppellten Trennstrich von den restlichen Einträgen getrennt.

user	user's identifier	number
group	user's group identifier	number
role	user's role identifier	number
createdBy	-	number
createdSerial	-	number

Tabelle 4.1: Tabelle m

uid	user's identifier	number
dn	DN of the user's certificate (Subject)	text
caid	Identifier of the user's CA	number
cn	User's CN	text
mail		text
cauri		text
createdBy		number
createdSerial		number

Tabelle 4.2: Tabelle users

Die einzelnen Tabellen sind über die Primärschlüssel miteinander verknüpft. Dies wird an folgender Skizze noch einmal verdeutlicht:

caid	Certification Authority identifier	number
cadn	CA Subject	text
cadescr	Name of Certification Authority	text

Tabelle 4.3: Tabelle ca

aclid	acl identifier	number
principal	DN of administrator's certificate	number
operaion		number
allow		boolean
createdBy		number
createdSerial		number

Tabelle 4.4: Tabelle acl

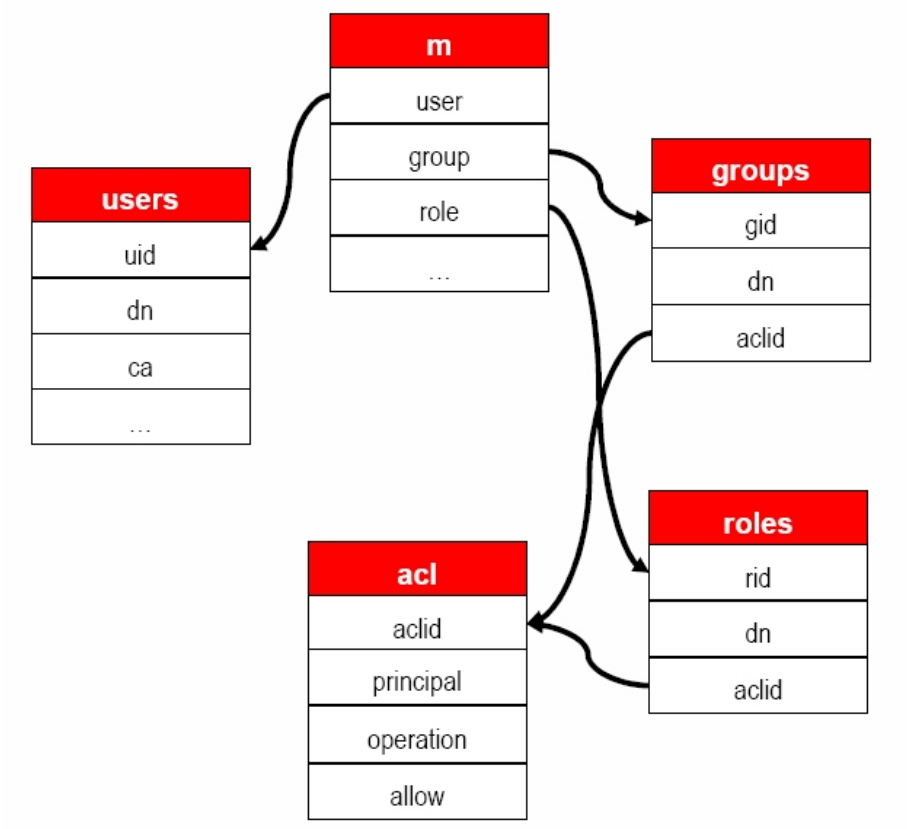

Abbildung 4.4: Verknüpfungen zwischen den Tabellen, aus [vom b]

4.4 Erstellen eines Proxy-Zertifikats

Möchte ein Benutzer auf die Ressourcen im Grid Zugriff erhalten, benötigt er dafür ein von VOMS ausgestelltes Zertifikat. Um dieses zu erhalten, muss erst eine Anfrage an den VOMS-Server gestellt werden, welcher dann die dem Benutzer zugeschriebenen Rechte an den VOMS-Client zurückschickt, woraufhin dieser dann das Zertifikat erstellen kann.

In der Abbildung wird der Ablauf dargestellt, der zum Erstellen eines temporären Proxy-Zertifikats führt, im Folgenden soll er noch mal im Detail vorgestellt werden:

gid	group identifier	number
dn	group name	text
aclid	the acl identifier of the administrator	number
createdBy		number
createdSerial		number

Tabelle 4.5: Tabelle groups

rid	role identifier	number
dn	role name	text
aclid	the acl identifier of the administrator	number
createdBy		number
createdSerial		number

Tabelle 4.6: Tabelle roles

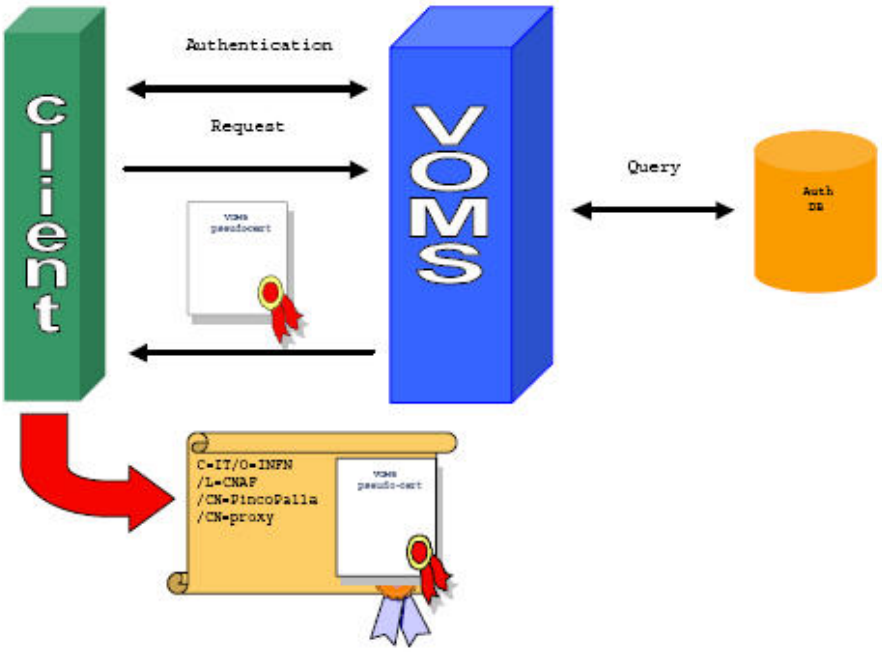

Abbildung 4.5: Vorgehen beim Erstellen eines Zertifikats, aus [Alfi]

1. Client und VOMS-Server authentisieren sich gegenseitig anhand ihrer Zertifikate. Mittels Standard Globus-API wird ein sicherer Kommunikationskanal eingerichtet.
2. Der Client sendet die Anfrage an den VOMS-Server.
3. Der VOMS-Server identifiziert den Benutzer und prüft die Anfrage auf ihre syntaktische Korrektheit.
4. Der VOMS-Server sendet die geforderten Informationen an den Client. Die Informationen sind die Rechte, die der Benutzer im Grid hat. Sie werden in einem RFC 3281 Attributs-Zertifikat übertragen, welches vom Server signiert wird.
5. Der Client prüft die erhaltenen Informationen auf ihre Korrektheit.
6. Hat der Benutzer die Rechte aufgrund seiner Mitgliedschaft in mehreren VOs beantragt, so arbeitet der Client die vorhergehenden Schritte für jeden VOMS-Server einzeln ab.

7. Der Client kann nun das temporäre Proxy-Zertifikat erstellen und die erhaltenen Informationen von allen VOMS-Servern einfügen.
8. Optional können nun noch weitere Authentifizierungsinformationen, wie z.B. Kerberos Tickets an das Zertifikat angefügt werden.

Kapitel 5

Installation und Bedienung von VOMS

5.1 Installation der erforderlichen Softwaremodule

Die Installation von VOMS setzt einige Softwarekomponenten voraus. Für den VO Client, der dann im Wesentlichen aus dem Aufruf von `grid-proxy-init` besteht, gibt es keine Voraussetzungen.
Für den VO-Server müssen dagegen mehrere Softwaremodule installiert werden. Zum einen wird eine Datenbank benötigt. Man hat die Wahl zwischen MySQL oder Oracle. Zur Administrierung einer VO gibt es ein Webinterface, was eine Tomcat Installation voraussetzt. Für die Installation des Admin-Interface wird außerdem eine Perl 5 Version vorausgesetzt. Zusätzlich sind etliche weitere Software-Pakete notwendig. Darunter sind einige Erweiterungen für Perl, um z.B. eine verschlüsselte Kommunikation im Netzwerk zu ermöglichen.
Empfohlen wird an dieser Stelle eine möglichst aktuelle RedHat oder Suse Installation. Bei älteren Versionen gibt es mögliche Konflikte, insbesondere bei Perl. Perl wird zur VO Administrierung benötigt und eine alte Version führt bereits bei der Installation zu Problemen. Pakete, die zur VOMS Installation notwendig sind, lassen sich in der aktuellen Version nur mit einer aktuellen Perl Version, mindestens 5.8.1, installieren. Eine ältere Perl Versionen erschwert also bereits die Installation der erforderlichen Pakete. Je nach RPM Packet der Admin Komponente müssen noch Anpassungen an die jeweilige Linux-Distribution getätigt werden. Dies reicht von Kleinigkeiten wie Pfadangaben bis hin zum Abändern von Distributionsspezifischen Consolenbefehlen.

5.2 Installation von VOMS

Auf den Clientsystemen muss lediglich das VOMS-Client Paket installiert werden. Hier wurde voms-client_gcc3_2_2-1.5.4-1_sl3 verwendet.

Wird ein VOMS-Server aufgesetzt, so sind folgende Pakete für den Server selbst erforderlich (hier aus einer glite-Installation [gli]):

- glite-security-voms-admin-client-1.0.7-1
- glite-security-voms-admin-interface-1.0.2-1
- glite-security-voms-admin-server-1.1.2-1
- glite-security-voms-api-1.5.9-0
- glite-security-voms-api-c-1.5.9-2
- glite-security-voms-api-cpp-1.5.9-0
- glite-security-voms-clients-1.5.9-0
- glite-security-voms-config-1.5.9-0
- glite-security-voms-mysql-1.0.3-0

- glite-security-voms-server-1.5.9-0
- glite-voms-server-config-2.0.0-3

Im Anhang findet sich die komplette Liste der Pakete die bei einer glite-Installation erforderlich sind.

Zusätzlich werden je nach System noch weitere Pakete benötigt. Neben Paketen wie 4Suite müssen verschiedene Perl Module nachinstalliert werden, so z.B. das Perl-XML Modul um den Einsatz von XML zu ermöglichen.

Zusätzlich wird eine OpenSSL Installation vorausgesetzt.

5.3 Erstellen einer VO

Nach der Installation der einzelnen Softwarepakete kann man bereits eine VO erstellen. Die erforderlichen Tabellen in der MySQL Datenbank werden durch den Aufruf von `/opt/edg/libexec/voms/voms_install_db` angelegt. Zusätzlich werden dabei noch einige Dateien für die VO erstellt. Darin werden Informationen zur VO gespeichert, unter anderem der Port, unter dem der passende VO Server erreichbar ist und die ID des Servers. Auch die Daten, die für den Zugriff auf die Datenbank notwendig sind, werden hier abgelegt. Eine genau Auflistung und Beschreibung der Dateien kann man sich hier anschauen.

Der Aufruf benötigt die Angaben von verschiedenen Parametern:

-mysql-home Zur Angabe des Home Verzeichnisses von MySQL

-db Name der Datenbank, in der die Informationen zur VO gespeichert werden. Standard ist hier „voms“. Muß nur geändert werden, wenn mehrere Server auf einem Rechner betrieben werden.

-port Port, auf dem der VOMS-Server erreichbar sein wird. Es gibt keinen Standardwert. Empfohlen wird, den ersten Server auf Port 15000 zu betreiben und an die folgenden jeweils einen höheren Port zu vergeben.

-voms-vo Name der VO, zu der der VOMS-Server gehört. Auch hier gibt es keinen Standardwert, so dass dieser immer mit angegeben werden muß.

-mysql-admin Name des MySQL Root Benutzers. Notwendig, um eine neue Datenbanktabelle und Benutzer in der Datenbank anzulegen. Standard ist hier „root“.

-mysql-pwd Passwort des Root Benutzers

-voms-name Benutzername des VOMS MySQL Accounts. Unter diesem Benutzernamen ist der Zugriff auf die neu erstellte Datenbank möglich.

-voms-pwd Passwort für den Benutzer, der durch „-voms-name“ erstellt wurde.

-code auf diesem Rechner eindeutige ID des VOMS-Servers. Wert liegt zwischen 0 und 65535. Standard ist hier 0.

Das Anlegen des ersten VO Servers könnte also folgendermaßen aussehen:

```
voms_install_db -port 15000 -voms-vo my-vo -mysql-pwd secret -voms-name
admin -voms-pwd secret
```

Einen weiteren VO Server könnte man so anlegen:

```
voms_install_db -port 15001 -voms-vo second-vo -mysql-pwd secret
-voms-name admin -voms-pwd secret -code 1
```

5.4 Administration einer VO

Ist eine VO erstellt worden, so kann der Administrator die weitere Konfiguration per Webinterface vornehmen. Dazu muss aber zuerst ein Benutzer als Administrator für die VO eingerichtet werden. Dies kann auf zwei verschiedene Arten erfolgen:

`edg-voms-admin --url=...<Name der VO> create-user usercert.pem assign-role <Name der VO> VO-Admin ucercert.pem` Wenn der VO Server auf dem lokalen Rechner läuft, dann würde die URL lauten: `http://localhost:8080/edg-voms-admin/`

oder alternativ mit dem lokalen Aufruf: `edg-voms-admin-local <Name der VO> --add-admin usercert.pem`

Ist der Benutzer als Administrator eingerichtet, so kann er auf das Webinterface zugreifen. Wenn dieses auf dem lokalen Rechner läuft, ist es über folgende URL erreichbar: `https://localhost:8443/edg-voms-admin/<Name der VO>`

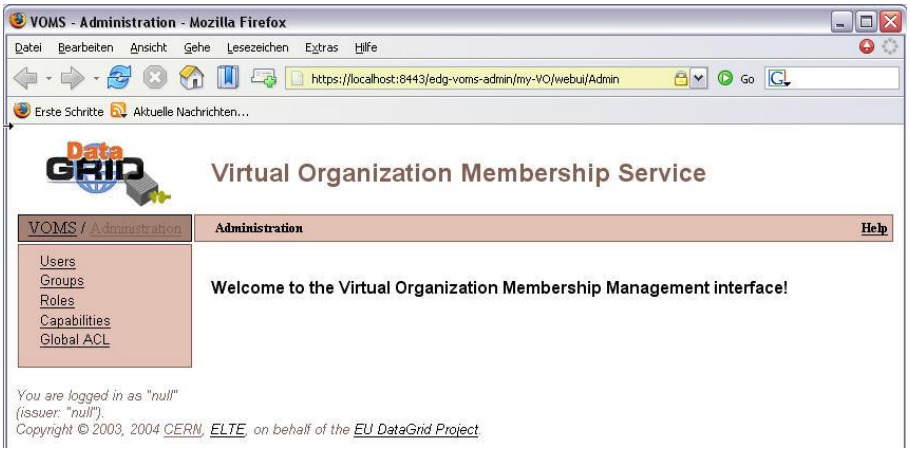

Abbildung 5.1: VOMS Webinterface

Über das Webinterface können neue Benutzer hinzugefügt und Benutzer, die eine Virtuelle Organisation verlassen haben, wieder entfernt werden. Gruppen und Rollen können definiert werden und mit Zugriffsrechten auf Services und Ressourcen im Grid ausgestattet werden. Jegliche Änderungen werden direkt in die Datenbank geschrieben. Der vorhergehende Zustand der Datenbank bleibt aber im Hintergrund erhalten und wird in weiteren Tabellen abgelegt. Dadurch ist es später möglich, einen Zustand zu einem früheren Zeitpunkt nachzuvollziehen. So kann z.B. ermittelt werden, ob ein bestimmter Benutzer zum gefragten Zeitpunkt überhaupt Zugriffsrechte auf einen Service hatte.

5.5 Bedienung aus Benutzersicht

Bei der Konzeption von VOMS wurde bedacht, dass sich für den Benutzer möglichst wenig ändern sollte. Der Benutzer muss aber natürlich weiterhin ein Proxy-Zertifikat erstellen. Statt der bisherigen Benutzung von `grid-proxy-init` erhält er ein Zertifikat durch den Aufruf von `voms-proxy-init`. Es wird in beiden Fällen ein Benutzerzertifikat erstellt, nur enthält es durch den neuen Aufruf zusätzlich Benutzerinformationen die vom VOMS-Server bereitgestellt wurden. Somit erhält der Benutzer aufgrund des Zertifikats Zugriff auf die Ressourcen im Grid, für die er eine Berechtigung hat. Dem Aufruf muss mindestens der Parameter `--voms <server[:command]>` folgen. Damit kann der Benutzer angeben, bei welcher VO er sich authentifiziert und somit auf welche Ressourcen er dann durch sein Zertifikat Zugriff erhalten wird. Standardmäßig werden alle Rechte, die der Benutzer in der VO hat, vom VOMS-Server

übertragen. Der Benutzer kann dies jedoch durch `command` einschränken. Gehört der Benutzer z.B. zu mehreren Gruppen innerhalb der VO, benötigt im Moment aber nur die Rechte, die sich aus der Gruppenzugehörigkeit A ergeben, so kann er dies durch den Befehl G<A> angeben.

5.6 Zugehörigkeit zu mehreren Virtuellen Organisationen

Ist der Benutzer Mitglied bei mehreren Virtuellen Organisationen, so kann er dies beim Authorisieren im Grid angeben. Dadurch bekommt er alle Rechte, die sich aus seinen Zugehörigkeiten ergeben. Dazu kann der `--voms` Parameter mehrmals hintereinander verwendet werden. Die Anfragen an die jeweiligen VOMS-Server werden nacheinander abgearbeitet. Die dabei entstehenden Pseudo-Zertifikate werden dann vom Client zu einem gemeinsamen Zertifikat zusammengefügt. An dieser Stelle können auch noch weitere Informationen in das Zertifikat eingebunden werden, z.B. ein bereits vorhandenes Kerberos Ticket. Als weitere Parameter stehen beim Aufruf von `voms-proxy-init` zur Verfügung:

-print Gibt die Informationen die vom Server geliefert werden, auf dem Bildschirm aus, anstatt ein Zertifikat zu erstellen.

-vomslife <**num**> Lebenszeit des Zertifikats in Stunden. Kann nur kürzer sein, als die vom Server standardmäßig gesetzte Lebenszeit.

-include <**file**> Einbinden von weiteren Dateien in das Proxy-Zertifikat, z.B. ein Kerberos Ticket

-order <**group**> Gruppen werden vom Server in der gewünschten Reihenfolge geliefert

Kapitel 6

Zusammenfassung und Bewertung

Für den Benutzer ändert sich nach der Einrichtung von VOMS wenig. Statt sich im Grid über das durch `grid-proxy-init` erstellte Zertifikat zu authorisieren, beantragt er sein Zertifikat mit `voms-proxy-init` und erhält dadurch den Zugriff auf die ihm zustehenden Ressourcen. Dabei muss sich der Benutzer nicht darum kümmern, wo diese Berechtigungen gespeichert werden. Außerdem ist das Vorgehen beim Authorisationsvorgang unabhängig von der Strukturierung der einzelnen Virtuellen Organisationen, zu denen der Benutzer gehört. Auch bei der Zugehörigkeit zu mehreren VOs wird der gesamte Vorgang durch einen einzigen Aufruf durchgeführt. Es müssen lediglich die einzelnen VOs als Parameter übergeben werden.

Ist VOMS im Grid installiert, so kann der Administrator einer VO alle anfallenden Änderungen an der VO bequem über ein Webinterface durchführen. Dafür ist also auch keine physische Anwesenheit erforderlich. Änderungen per remote-Zugriff sind also problemlos möglich. Auch können Rechte deligiert werden, indem weitere Benutzer in den Admin Status erhoben werden.

Das Problem könnte lediglich darin bestehen, eine erfolgreiche VOMS Installation durchzuführen. Neben den einfach erfüllbaren Voraussetzungen, eine Datenbank und einen Webserver einzurichten, werden viele system-erweiternde Pakete gefordert. Dabei kann es schwierig sein, diese in aufeinander abgestimmten Versionen zu installieren. Auch kann es gerade bei bereits länger existierenden Grids Schwierigkeiten bereiten, VOMS auf einer alten Linux-Distribution zu installieren. Dabei ist dann ein Update des Systems empfehlenswert, was natürlich unter Umständen kompliziert und aufwendig werden kann.

Wünschenswert für die Zukunft wäre eine Möglichkeit, Mitgliedsinformationen zwischen verschiedenen VOMS Servern auszutauschen. Gerade bei größeren Grids und komplexeren Aufgabenbereichen kann es erforderlich sein, dass Benutzer zu Mitgliedern in verschiedenen Virtuellen Organisationen werden. Eine Art Vererbung würde dann die Einrichtung und Verwaltung erleichtern. Ein Wechsel der Mitgliedschaft aus einer VO in eine andere wäre über eine Art von Export- bzw. Importfunktion wünschenswert.

Ein weitere wünschenswerte Erweiterung wäre es, auch Diensten den Zugriff auf Ressourcen im Grid zu erlauben. Aktuell muss ein Dienst unter dem Namen einer realen Person ausgeführt werden, weil eine Authorisierung nur aufgrund eines Zertifikats erfolgen kann. Dieses wird von den Certificate-Authorities momentan nur an reale Personen ausgestellt.

Neben den aufgeführten Punkten hat VOMS noch einige weitere Schwächen. Aus der Bemühung heraus, diese zu beseitigen entstand das VOX-Projekt (VO Management Service eXtension). Im Rahmen dieses Projektes entstand VOMRS [vom a] und es kann als eine Art Weiterentwicklung von VOMS gesehen werden. Zielsetzung war hierbei ein System zu entwickeln, welches Policy-basierend die Zugehörigkeit von Mitarbeitern zu einer Virtuellen Organisation ausdrückt. Darauf aufgbauend sollten dann die Zugriffsrechte im Grid geregelt und Zugriffe auch überwacht werden können.
Ähnlich wie bei VOMS besteht auch dieses System aus einer Server und einer Client Komponente. Dem Admin steht hierbei auch ein Webinterface zur Verfügung.

Besonderes Augenemerk wurde auf Mängel bei VOMS gelegt, die bei kleinen Virtuellen Organisationen u.U. nicht so gewichtig sind, jedoch bei großen VOs sehr störend sein können. So erfordert z.B. der Beitritt neuer Benutzer zu einer VO bei VOMS relativ viel Interaktion des Admins. Bei VOMRS wird dies effizienter gelöst so dass dem Admin Arbeit abgenommen wird.
VOMRS bietet im Gegensatz zu VOMS die Möglichkeit, mehr Informationen zu einzelnen Benutzern zu speichern, wie z.B. die Telefonnummer oder auch beliebige andere Angaben.
Wird einem Benutzer die Zugehörigkeit zu einer Virtuellen Organisation nur vorübergehend entzogen, so gibt es bei VOMS nur die Möglichkeit den Benutzer komplett aus der VO zu entfernen so dass er bei späterem Wiedereintritt wieder die komplette Registrierung bei der VO durchlaufen muss und alle Angaben erneut getätigt werden müssen. VOMRS bietet hier die Möglichkeit, Benutzer vorübergehend zu suspendieren und sie zu einem späteren Zeitpunkt wieder zu resumen, so dass die dann wieder über alle Rechte verfügen.

Anhang A

Liste der erforderlichen Pakete zur VOMS Installation

```
Das glite VOMS-Paket ist hier zum download verfügbar:
http://glite.web.cern.ch/glite/packages/R1.5/R20051130/default.asp

Es enthält folgende Dateien, die zur Installation von VOMS benötigt werden:

MySQL-client-4.1.11-0.i386.rpm
MySQL-server-4.1.11-0.i386.rpm
glite-config-1.6.22-1.noarch.rpm
glite-security-trustmanager-1.8.2-1.noarch.rpm
glite-security-util-java-1.3.0-1.noarch.rpm
glite-security-voms-admin-client-1.2.10-1.noarch.rpm
glite-security-voms-admin-interface-1.0.2-1.noarch.rpm
glite-security-voms-admin-server-1.2.10-1.noarch.rpm
glite-security-voms-api-1.6.10-0.i386.rpm
glite-security-voms-api-c-1.6.10-0.i386.rpm
glite-security-voms-api-cpp-1.6.10-0.i386.rpm
glite-security-voms-clients-1.6.10-0.i386.rpm
glite-security-voms-config-1.6.10-0.i386.rpm
glite-security-voms-mysql-1.1.2-0.i386.rpm
glite-security-voms-server-1.6.10-0.i386.rpm
glite-voms-server-mysql-config-2.2.5-1.noarch.rpm
gpt-VDT1.2.2rh9-1.i386.rpm
perl-Authen-SASL-2.08-1.1.el3.rf.noarch.rpm
perl-Crypt-SSLeay-0.51-4.i386.rpm
perl-Digest-HMAC-1.01-11.1.noarch.rpm
perl-Digest-SHA1-2.01-15.1.i386.rpm
perl-MIME-Lite-2.117-2.1.el3.rf.noarch.rpm
perl-Net-Jabber-2.0-1.1.el3.rf.noarch.rpm
perl-Net-XMPP-1.0-1.1.el3.rf.noarch.rpm
perl-SOAP-Lite-0.60a-1.1.el3.rf.noarch.rpm
perl-XML-Stream-1.22-1.1.el3.rf.noarch.rpm
tomcat5-5.0.28-11_EGEE.noarch.rpm
vdt_globus_essentials-VDT1.2.2rh9-1.i386.rpm

****************** /glite-rgma-servicetool: ******************
glite-config-1.6.22-1.noarch.rpm
glite-essentials-java-1.2.0-2_EGEE.noarch.rpm
glite-rgma-api-java-5.0.3-1.noarch.rpm
```

```
glite-rgma-base-5.0.4-1.noarch.rpm
glite-rgma-common-config-5.0.1-1.noarch.rpm
glite-rgma-servicetool-5.0.3-1.noarch.rpm
glite-rgma-servicetool-config-5.2.2-1.noarch.rpm
glite-rgma-stubs-servlet-java-5.0.3-1.noarch.rpm
glite-security-trustmanager-1.8.2-1.noarch.rpm
glite-security-util-java-1.3.0-1.noarch.rpm
glite-security-utils
glite-security-utils_installer.sh

****************** /glite-security-utils: ******************
ca_AIST-1.0-1.noarch.rpm
ca_ASGCCA-1.0-1.noarch.rpm
ca_ArmeSFo-1.0-1.noarch.rpm
ca_AustrianGrid-1.0-1.noarch.rpm
ca_BEGrid-1.0-1.noarch.rpm
ca_BalticGrid-1.0-1.noarch.rpm
ca_CERN-1.0-1.noarch.rpm
ca_CESNET-1.0-1.noarch.rpm
ca_CESNET-old-1.0-1.noarch.rpm
ca_CNRS-1.0-1.noarch.rpm
ca_CNRS-DataGrid-1.0-1.noarch.rpm
ca_CNRS-Grid-FR-1.0-1.noarch.rpm
ca_CNRS-Projets-1.0-1.noarch.rpm
ca_CyGrid-1.0-1.noarch.rpm
ca_CyGrid-old-1.0-1.noarch.rpm
ca_DFN-GridGermany-Root-1.0-1.noarch.rpm
ca_DFN-GridGermany-Server-1.0-1.noarch.rpm
ca_DFN-GridGermany-User-1.0-1.noarch.rpm
ca_DOEGrids-1.0-1.noarch.rpm
ca_ESnet-1.0-1.noarch.rpm
ca_EstonianGrid-1.0-1.noarch.rpm
ca_GermanGrid-1.0-1.noarch.rpm
ca_Grid-Ireland-1.0-1.noarch.rpm
ca_GridCanada-1.0-1.noarch.rpm
ca_HellasGrid-1.0-1.noarch.rpm
ca_IHEP-1.0-1.noarch.rpm
ca_INFN-1.0-1.noarch.rpm
ca_IUCC-1.0-1.noarch.rpm
ca_KISTI-1.0-1.noarch.rpm
ca_LIP-1.0-1.noarch.rpm
ca_LIPCA-1.0-1.noarch.rpm
ca_NIIF-1.0-1.noarch.rpm
ca_NIKHEF-1.0-1.noarch.rpm
ca_NorduGrid-1.0-1.noarch.rpm
ca_PK-Grid-1.0-1.noarch.rpm
ca_PolishGrid-1.0-1.noarch.rpm
ca_RDIG-1.0-1.noarch.rpm
ca_RMKI-1.0-1.noarch.rpm
ca_Russia-1.0-1.noarch.rpm
ca_SEE-GRID-1.0-1.noarch.rpm
ca_SWITCH-1.0-1.noarch.rpm
ca_SiGNET-1.0-1.noarch.rpm
ca_SlovakGrid-1.0-1.noarch.rpm
ca_Spain-1.0-1.noarch.rpm
ca_SwissSign-Bronze-1.0-1.noarch.rpm
ca_SwissSign-Root-1.0-1.noarch.rpm
ca_SwissSign-Silver-1.0-1.noarch.rpm
```

```
ca_TRGrid-1.0-1.noarch.rpm
ca_UKeScience-1.0-1.noarch.rpm
ca_list
ca_policy_eugridpma-1.0-1.noarch.rpm
ca_policy_eugridpma-classic-1.0-1.noarch.rpm
ca_policy_igtf-classic-1.0-1.noarch.rpm
ca_policy_igtf-slcs-1.0-1.noarch.rpm
edg-mkgridmap-2.5.1-1_sl3.noarch.rpm
edg-utils-system-1.7.0-1.noarch.rpm
glite-config-1.6.22-1.noarch.rpm
glite-security-utils-config-1.2.2-1.noarch.rpm
perl-Convert-ASN1-0.18-0.rhel3.dag.noarch.rpm
perl-Crypt-SSLeay-0.51-4.i386.rpm
perl-IO-Socket-SSL-0.94-0.dag.rhel3.noarch.rpm
perl-Net-LDAP-0.2701-1.dag.rhel3.noarch.rpm
perl-Net-SSLeay-1.23-0.dag.rhel3.i386.rpm
perl-TermReadKey-2.20-12.i386.rpm
```

Abkürzungsverzeichnis

CA Certificate-Authority

CERN European Organization for Nuclear Research

EGEE Enabling Grids for E-SciencE

ISSRG Information Systems Security Research Group

MACE Middleware Architecture Committee for Educationist

SAML Security Assertion Markup Language

VO Virtuelle Organisation

VOMS Virtual Organization Membership Service

Web-SSO Web Single SignOn

XML Extensible Markup Language

Literaturverzeichnis

[Alfi] ALFIERI, R.: *VOMS, an Authorization System for Virtual Organizations*, http://grid-auth.infn.it/docs/VOMS-Santiago.pdf .

[cas] *CAS Homepage*, http://www.globus.org/grid_software/security/cas.php .

[cer] *CERN Homepage*, http://www.cern.ch .

[d-g] *D-Grid Homepage*, http://www.d-grid.de .

[EGE] *EGEE Homepage*, http://www.eu-egee.org .

[fop] *Fortgeschrittenenpraktikas am Lehrstuhl für Kommunikationssysteme und Systemprogrammierung der LMU*, http://www.nm.ifi.lmu.de/pub/Fopras/ .

[ga] *Globus Alliance Homepage*, http://www.globus.org/alliance .

[gli] *gLite Homepage*, http://glite.web.cern.ch .

[gri] *GridShib Homepage*, http://gridshib.globus.org .

[gt] *Globus Toolkit Homepage*, http://www.globus.org/toolkit/ .

[lib] *Project Liberty Homepage*, http://www.projectliberty.org .

[per] *Permis Homepage*, http://sec.isi.salford.ac.uk/permis/ .

[shi] *Shibboleth Homepage*, http://shibboleth.internet2.edu .

[sun] *Sun Access Manager Homepage*, http://www.sun.com/software/products/access_mgr .

[vom a] *VOMRS Homepage*, http://www.uscms.org/SoftwareComputing/Grid/VO/ .

[vom b] *VOMS Architecture v1.1*, http://grid-auth.infn.it/docs/VOMS-v1_1.pdf .

[vom c] *VOMS Homepage*, http://hep-project-grid-scg.web.cern.ch/hep-project-grid-scg/voms.html .