Federated Authorisation and Group Management in e-Science

Summary
The power of federated identity and access management is not just simplified authentication but the ability to make authorisation decisions based on a range of attributes. In an increasingly inter-federated world, how can we present and manage authorisations across different federated services in a consistent and uniform way with minimal overhead for each of the parties involved (service providers, identity providers, users)? If this would be done through a centralised service, which party would offer this service? How to assure unambiguous attribute interpretation in and across federations? How to manage groups of users that do not always fit in the same organisational mould? Does the user have a say in the release of attributes?

These and other authorisation-related questions are addressed in this deliverable. It presents a state-of-the-art overview of authorisation standards, models and technologies. The overview is presented in the context of e-Science where researchers from different institutions need to obtain access to resources and share resources with each other in collaborating groups. Extra attention is therefore paid to the relation between group management and authorisation.

The deliverable concludes with challenges and recommendations for future authorisation research in GigaPort3 and adoption in the SURFfederatie.
This project was made possible by the support of SURF, the collaborative organisation for higher education institutes and research institutes aimed at breakthrough innovations in ICT. More information on SURF is available on the website [www.surf.nl](http://www.surf.nl).
Contents

1 Introduction 1
   1.1 Scope 1
   1.2 Motivation 2
   1.3 Objective 4
   1.4 Reading guide 4

2 Authorisation use cases and models 5
   2.1 Authorisation use cases 5
      2.1.1 Progress test Feedback system 5
      2.1.2 LMS 5
      2.1.3 Sharing of experimental data 5
      2.1.4 Cross-domain light-paths 6
      2.1.5 Age-based access 6
      2.1.6 SP-specific attributes 6
      2.1.7 Local authorisation hassle 6
      2.1.8 General analysis and requirements 7
   2.2 General authorisation abstractions 8
      2.2.1 Authorisation policies 9
         2.2.1.1 Role Based Access Control (RBAC) 9
         2.2.1.2 RBAC in distributed environments 10
      2.2.2 Authorisation models 12
         2.2.2.1 Entities 12
         2.2.2.2 Sequences 12
         2.2.2.3 Tokens 13
         2.2.2.4 Functional architecture 15
         2.2.2.5 Process-view 16
      2.2.3 Authorisation mechanisms 17
         2.2.3.1 Externalising authorisation 17
         2.2.3.2 Federated authorisation 19
   2.3 Summary 23

3 Policy-based authorisation solutions 24
   3.1 Web-based 24
      3.1.1 XACML 24
         3.1.1.1 Overview 24
         3.1.1.2 In federations 27
         3.1.1.3 Interoperability and profiles 27
      3.1.2 SAML-based authorisation 28
      3.1.3 WS-* 29
         3.1.3.1 WS-Policy 29
         3.1.3.2 WS-SecurityPolicy 29
   3.2 User controlled 30
      3.2.1 OAuth 30
      3.2.2 User-Managed Access 33
      3.2.3 Lockr 35
      3.2.4 Invitation-based authorisation 36
   3.3 Grid-based 37
3.3.1 OGF recommendations 37
3.3.2 Globus Toolkit 38
3.3.3 Akenti 39
3.3.4 SecPAL 40

3.4 PMI-based 40
3.4.1 Signet 41
3.4.2 PERMIS 42

3.5 Summary 43

4 Group-based authorisation 45
4.1 Authorisation in group management solutions 47
4.1.1 CAS 47
4.1.2 VOMS 48
4.1.3 G-Box 49
4.1.4 Grouper 51

4.2 Group attribute-based access 52
4.3 Summary 53

5 Authorisation challenges in federations 55
5.1 Generic challenges 55
5.1.1 Heterogeneity of systems 55
5.1.1.1 OpenAz 56
5.1.2 Discovery of PDPs, policies and attributes 57
5.1.3 Communication performance 58
5.1.4 Session-based authorisation 59
5.1.5 User oriented authorisation 59
5.1.6 Groups and collaborations 59
5.1.7 Access control to authorisation decisions 60

5.2 ABAC specific challenges 60
5.2.1 Understanding attributes - formats and semantics 61
5.2.2 Attribute assurance - Trustworthiness of attributes 63
5.2.3 Attribute sharing - ArisID 64
5.2.4 Conditional replies and obligations 65

5.3 Policy management related challenges 66
5.3.1 Policy integration and interoperability 66
5.3.2 Policy conflict resolution 68
5.3.3 Policy modularity and reusability 68
5.3.4 Policy provisioning 68
5.3.5 Policy creation and analysis 68

5.4 Summary 69

6 Conclusions 71
6.1 Role of SURFnet and GigaPort3 72

References 75
1 Introduction

1.1 Scope

Modern e-Science infrastructures allow researchers the flexibility of iterative workflows, dynamic service composition, ease of collaborative data management, and insurance that data remains well-curated and publication-ready with appropriate metadata, provenance, and authorisation. The latter aspect of authorisation is certainly one of the key elements contributing to the success of e-Science and will be the focus of this deliverable.

To enable access to research sources and services, authentication is first required to establish the identity of the user. Once a user has been identified, authorisation is required to determine what rights should be given to the user. These rights are often based on the role of the user in their own organisation, for example, researcher. In other words, authorisation provides the definitions of what resources researchers can actually access once they are authenticated (i.e. their identity is established).

Collaboration and sharing of resources between e-Scientists has, up to now, been enabled by the creation of custom user accounts on machines and services, with access and protection typically given by digital certificates or username/password combinations [1]. This inevitably leads to the situation where a user holds multiple user accounts on separately administered resources. As the number of e-Science users grows, this decentralised model becomes unmanageable and simply does not scale.

Federated identity and access management offers a solution by having a federation of identity providers (IdPs) and service providers (SPs) that trust each other. With federated identity and access management, the researcher can use his local institution’s account to access services or resources offered by other institutions or commercial SPs without having the need to create an account or login again. It allows cross-organisational single sign-on to trusted services offered by peer institutions within the federation. In e-Science, federated identity and access management allows researchers to share their valuable resources (data, equipment, articles, computing power) with fellow e-Scientists in other institutions in the same federation or even across federations.
Today's federated identity solutions are mostly concerned with authentication and only rudimentary authorisation. Particularly, more flexible and multi-domain authorisations are needed to support emerging e-Science scenarios [2]. A major issue in an identity federation is related to the management of access rights to federated resources. These rights are typically specified in policies. In e-Science federations or so-called Virtual Organisations (VOs), researchers and SPs will need to have a unified, conceptually centralised “view” of the policies that they have specified and a unified understanding of how the policies will play out in the underlying infrastructure. This understanding can for instance be based upon roles or attributes; the latter seems the most flexible for use in a federated e-Science environment.

Group management is essential in any e-Science environment. Groups provide the flexibility for managing access and offering collaborative services. Groups can be generated based on an attribute about a party, or groups/VOs can be formed ad hoc, when parties need to collaborate and share resources. Identifying the groups or VOs that a person belongs to can be essential to defining which resources a party can access. This allows better granularity for service authorisations and helps with group management for access to e.g. wikis, blogs and other collaborative software.

A typical use case that illustrates the need for efficient authorisation and group management is a PhD student who needs to prove that:

- Studielink says she is a student;
- A particular faculty says she is part of their team;
- The university says she is part of their faculty.

Each of these authorities could make a claim about the identity of the student by asserting some identity-related attributes. The required claims or attribute-assertions, however, are specified in an authorisation policy and need to be enforced prior to granting access to a resource. The policy specification is done by an appropriate authority. This can be either the resource owner (which is the common case) or a trusted third party.

In e-Science, the researcher often is the resource owner. He should be given control over whom he will give access to his resources. The researcher somehow must be facilitated in specifying authorisation policies that allow colleague researchers access to a certain set of resources without having to give away his own account credentials.

### 1.2 Motivation

SURFnet’s SURFfederatie facilitates user identity information exchange, needed for authentication and authorisation, between different IdPs and SPs in the educational domain. Though the authentication part has been prime focus in the past years, the authorisation capabilities have been less widely explored. This is acknowledged in a survey kept amongst the participants of the SURFfederatie (see Figure 1).
The authorisation process is a critical aspect in distributed and federated environments. For example, in federated identity management scenarios “distributed” decisions and authorisations are made by multiple parties involved in an interaction or e-Science activity. Ensuring that the remote party is making the right decisions and carrying out the correct authorisation processes, based on agreed policies is not trivial.

Moreover, for many disciplines e-Science is not simply supporting access to and use of high performance computing clusters for larger scale simulations. Instead many disciplines require user-oriented support for finer-grained security-driven collaborative models that span multiple administrative or organisational domains [4]. This in turn requires addressing amongst others issues such as user-oriented authorisation, attribute-based authorisation, delegation, group management, and policy management.

As e-Science is primarily project-based this imposes additional requirements on the flexibility of authorisations. Compared to the rather static information being managed by the IdP a more dynamic form of authorisations is required to facilitate easy use of services and dynamic resources such as light-paths for members of groups and VOs. In contrast to the more static and established federation context, groups and VOs are far more dynamic and loosely coupled. This imposes additional strains on the authorisation management of group members to resources.
1.3 Objective

The objective of this deliverable is to identify the challenges concerning authorisation in federated environments. Particular focus will also be on authorisation for groups in such environments and in the context of e-Science. The identified challenges will be related to the current state-of-the-art authorisation technologies and solutions. Gaps will be identified and possible solutions will be compared to the current SURFfederatie and analysed in terms of their possible usefulness to enhance the SURFfederatie, i.e. make it more scalable, more flexible and efficient, and suitable for e-Science applications.

1.4 Reading guide

The structure of this deliverable is organised as follows. The next section 2 goes into the basic essentials of authorisation. It discusses authorisation use cases and models. Section 3 presents the state-of-the-art technology that is available to address authorisation in federated environments. More dynamic group aspects for resource access are described in section 4. The authorisation challenges in identity federations that can be distilled from the former sections are listed in section 5. Section 6 concludes with a summary of the main authorisation challenges and presents several recommendations for future work in GigaPort3.
2 Authorisation use cases and models

This section describes several illustrative authorisation and group use cases in a federative context and describes authorisation from different perspectives to come to a generic authorisation model.

2.1 Authorisation use cases

The following use cases exemplify the complexity of authorisation. The underlined text high-light the different authorisation aspects and challenges.

2.1.1 Progress test Feedback system

The Progress test Feedback system (ProF, [5]) enables students and staff to receive demand driven, longitudinal, web based feedback on the outcomes of the inter-university Progress Test in Medicine (iPGT). The progress test measures, at regular intervals, the knowledge of all students at graduation level. In this way information is obtained about the development of each medical student’s overall medical knowledge and the knowledge for different sub-domains.

Access to ProF is federated and granted via the SURFfederatie. Not only students and teachers but also staff members like advisors and mentors have access to the application. As is the case for many applications, the complex authorisation hierarchy that consists of several roles is implemented in the ProF application itself.

2.1.2 LMS

A Learning Management System (LMS) is typically used by teachers, students, and researchers in an institution of higher education and needs to deal with far more complex authorisation requirements than a hosted web application, a corporate site, or a government site. In higher education, a single individual may be an instructor in a classroom, a student in a seminar, a leader in a discussion group, a grader in a lab section, and an editor on a research project. Some of these roles and contexts (but not all) are typically determined outside the LMS. These context-dependent roles imply different application roles and permissions. The set of possible roles and their implied permissions vary from institution to institution, and even within an institution. The applications (and their own ideas of user functions) are developed independently and deployed over time.

2.1.3 Sharing of experimental data

A PhD researcher shares her experimental data with several researchers in other institutions in the federation. She defines a group of collaborating researchers and grants them access to her data and two additional web-based collaboration tools. Access to the data is based on group membership. When a student is going to support the PhD researcher, he is added to the group as well and immediately gains access to the experimental data and tools of that group.
2.1.4 Cross-domain light-paths

An employee at Company A wants to view a graphical resource at Company B. In order to view the resource properly, an end-to-end light-path between Company A and B needs to be established. A light-path application takes care of this by finding a possible path between the two companies and subsequent scheduling of the network resources that are offered by potentially multiple providers that constitute the path. The correct access credentials need to be provided by the application in order to access and schedule the light-path resources. These credentials are diverse and may consist of employee identity and role information, a proof of federation membership, and status entitlement (e.g. bronze, silver or gold).

2.1.5 Age-based access

A video-on-demand store represents adult customers by an attribute “fullAge” in its access control policy. Furthermore, an educational medical book vendor who is part of the same federation could describe adult customers using the attribute “age”. The authorisation query could be one of ‘What is the user’s birth date?’, ‘Is the user over 18?’ or ‘Can this user watch adult-rated content?’. On the other hand, a potential student customer might try to prove these properties by claiming that he has a driver’s license (“hasDriversLicence”) which can be valid for many countries.

2.1.6 SP-specific attributes

Sometimes an SP requires the IdP to provide a specific attribute for service access, e.g. a license-related attribute for downloading educational content. IdPs are then obliged to store these attributes in their LDAP databases. The advantage of such attributes is that it enables the IdP to only allow specific users access to those SP offerings thereby reducing the license costs considerably. Management of such SP-specific attributes, however, is time-consuming and contaminates the LDAP database. Couldn’t this be outsourced to or facilitated by the federation operator?

2.1.7 Local authorisation hassle

Since each application locally enforces its own authorisations, granting new users or denying fired users access to these applications requires updating many access control lists. Externalising authorisation functionality from the applications enables centralisation of it and subsequent more efficient access control (de)provisioning.
2.1.8 General analysis and requirements

The use cases show that access to resources is usually not for free. Users must somehow be able to present certain credentials to the resource provider in order to obtain access to these resources or to the services that control them. Resource providers will not always be interested in the true identity of the user. They do not know all the end-users and user account management is not an option as it is impossible for users to have an account at all possible resource providers. What resource providers are interested in are roles or credentials expressing a membership affiliation to a community or virtual organisation they have contracts with. It is up to the user to present these credentials (typically in the form of attributes or claims) in a secure, trusted, and efficient manner. An identity federation clearly helps to efficiently provide the required credentials in a trustworthy manner. Once the credentials are provided in a secure and trusted manner, access policy decisions have to be made based upon these credentials. A policy management, decision and enforcement infrastructure must be in place for this purpose.

Not only the resource or service provider should determine the user’s rights; the home institution of the user may also have a say in this. The identity provider is a potential source for additional policies for limiting access of its users to service providers, i.e. the identity provider may also have a say on what the user is entitled to do when accessing a service or resource. For instance that she only has read access. In that case the authorisation infrastructure should be able to combine multiple policies, i.e. from the identity and service provider.

Considering the converging attributes of federated collaborations and current trends of authorisation solutions as reflected in the use cases, the following requirements for authorisation in federations can be distilled [6]:

- Locality of control. Local authorities must be able to determine authorisation decisions according to their own criteria, and each collaborative partner must be able to control which parts of their policies to disclose to other collaborative partners.
- Compatibility with standards-based solutions. Use of open, standards-based modules ensures that a solution can be suitable for the majority of the federated scenarios. It also aids in scalable deployment without introducing significant changes to existing IT systems.
- Flexible choices of enforcement mechanisms. It is necessary that a decentralised authorisation model is not bound to a particular access control mechanism but rather enforceable by as many different types of enforcement mechanisms as possible.

In federated environments, authorisation policies for each administrative or organisational domain are determined by its local administrator or through cooperation of multiple administrators. When a user submits a request associated with a domain, possibly supported by one or more credentials, it must comply with the domain’s authorisation policies if it is to be granted [7]. Important other properties of authorisation in federated multi-domain environments include the following [8]:

[Autorisations and Group Management]
• Authorisation policies are created, stored, and managed in a dynamic and
distributed fashion by co-operation among several independent administrators
in order to e.g. support group-based access control.
• Accommodation of complex interactions which may involve authorisation
policies that utilise delegations, roles, and groups.
• Combination of multiple authorisation policies from different locations.
• Domains are typically heterogeneous; e.g. they have diverse restrictions.

These properties raise specific requirements for authorisation policy representation
in federated multi-domain environments including the following:
• More flexible, distributed, and declarative approach to authorisation
representation. It is most likely not possible to eliminate syntactical
translations of authorisation policies in the context of federated collaborations.
The number of translations required, however, can be reduced by utilising
standards-based specifications and by avoiding proprietary schemes.
• More expressive representation supporting heterogeneous or unknown
policies. It must be possible to express notions of roles, attributes,
capabilities, and temporal constraints must be able to be expressed.
• Scalability. Policy specifications must be enforced and managed by policy
administrators. Deployment of policies across multiple IdPs and SPs must be
simple and scalable.
• Robustness and unambiguity. Robust validation and analysis of policies is
required. Particularly when multiple policies from different domains need to be
combined and interpreted, interference of different authorisation policies must
be prevented.
• Extensibility. Changes in policies must be anticipated. It should be easy to add
new rules, entities, and conditions must be easily added or modified without
causing major changes on the structure of existing policies or the
infrastructure.

Policy specifications for federated systems may become very complex and error-
prone. This may be prevented by introducing (semantic) logic. In particular, logic
provides:
• Precise and non-ambiguous representation.
• Sufficient expressiveness, flexibility, and declarativeness for representation.
• Reasoning about authorisation policies and proving properties on them.
• Logical transformations of representations.

2.2 General authorisation abstractions

Authorisation can be discussed at different levels of abstraction. This includes
authorisation policies, mechanisms, and models. Policies describe how authorisation
is managed and who, under what circumstances, may access which resources.
Mechanisms enforce policies and define how access requests are evaluated against
those policies. Authorisation models bridge the gap between high-level policies and
low-level mechanisms by defining means of how authorisation rules should be
applied to protect resources. Such models are defined mostly in terms of subjects
and objects and possible interactions between them.
2.2.1 Authorisation policies

2.2.1.1 Role Based Access Control (RBAC)

Various authorisation policies have been proposed including mandatory, discretionary and role-based policies. In discretionary access control (DAC) policies control access based on the identity of the subject and on authorisation rules that define allowed operations on resources. Mandatory access control (MAC) policies control access based on centrally mandated sensitivity levels (classifications) of protected resources and authorisation levels of subjects (clearances). Role-based access control (RBAC) allows composing of authorisation policies that map naturally to an organisation’s structure. Access control decisions are made based on roles that individual subjects may possess and rules that are applied to resources.

RBAC was originally introduced to simplify the administration of access permissions, by avoiding direct assignment of them to individual users, and increasing the level of security by facilitating mechanisms to enforce Least Privilege and Separation of Duty principles.

Although RBAC is a well-defined model, most large RBAC projects turn out to be very costly and almost impossible to complete. The main issues with these projects are:

- Interoperability - difficulties to agree on a set of roles with a common meaning that can be shared between applications, platforms, domains and enterprises.
- Role explosion - too many roles without clear organisational definitions need to be administered. In some cases the number of roles has become larger than the number of users in the enterprise.

The interoperability issue is due to the fact that "role" is used in many different ways and there is no real consensus regarding the terminology. A role can refer to a job function within an organisational structure, a group and or a name for a collection of access permissions. However, a job function may have more than one name in different applications and domains, which of course, leads to confusion and a large number of non-interoperability issues.

The role explosion issue is due to the fact that RBAC - similar to the traditional Access Control List (ACL) policy approach - defines access permissions statically in the form of a snap-shot without considering the context of the access. Capturing the context, including the dynamics of the environment in which access permissions can be defined, would mean defining a large set of roles including permissions for each possible context. Moreover, defining fine grained access permissions would also create many sets of permissions causing role explosion.
2.2.1.2 RBAC in distributed environments

Both, RBAC and ACL are approaches that are applicable in enterprise environments. For decentralised or distributed settings, several variants of RBAC have been proposed:

- **DRBAC** - Distributed Role-Based Access Control (DRBAC) is a scalable, decentralised trust-management and access control mechanism for systems that span multiple administrative domains [9]. DRBAC utilises PKI identities to define trust domains, roles to define controlled activities, and role delegation across domains to represent permissions for these activities. The mapping of controlled actions to roles enables their namespaces to serve as policy roots. In DRBAC a set of distributed roles is defined, employed and shared by both resource provider and resource subscriber. In general, distributed roles are determined by a resource provider and the subscribing organisation will map its local roles to the distributed roles.

- **D-Role** - The DRBAC-derived distributed role (d-Role) concept provides another layer of abstraction between two different sets of roles from different organisational domains [10]. Through this indirection, roles can be mapped from public and private viewpoints of a participating organisation without causing changes in a global realm. Figure 2 illustrates an overview of how d-Role concept can bridge the organisational boundaries in multi-domain collaborations. In the figure, d-Role is further sub-classified as d-Role Provider and d-Role Requester, providing further distinction between a requester and a provider.

An extension of the eXtensible Access Control Markup Language (XACML, see also section 3.1.1) core specification to specify d-Roles has been proposed and implemented [11]. The set of privileges to a corresponding d-Role can be added as attribute fields of d-Role elements. The d-Role concept encapsulates the set of knowledge that needs to be shared among the collaborative partners.

![Distributed roles policy (XACML)](image)

Figure 2: Overview of decentralised authorisation with d-Role (from [6]).

- **X-GTRBAC** - XML-based Generalised Temporal Role Based Access Control (X-GTRBAC, [12]) is an XML-based policy specification language. The language conforms to the GTRBAC model that builds upon RBAC and supports temporal constraints on access control policies [13].
X-GTRBAC emphasises separation of language schemas to provide efficient specification of definitions of RBAC elements, user-to-role and permission-to-role assignments, hierarchical and separation of duty constraints, and an elaborate set of temporal constraints. X-GTRBAC can be used to specify GTRBAC policies for securing heterogeneous and distributed resources, and allowing dynamic evaluation of user credentials and context information to provide fine-grained access control. X-GTRBAC allows for centralised as well as decentralised policy administration. The decentralised model uses the notion of administrative domains to decentralise the policy administration tasks.

2.2.1.3 Attribute Based Access Control (ABAC)

In most identity federations, attribute-based access control (ABAC) is the preferred authorisation policy approach. In ABAC all aspects of an access request are considered and identified by attributes: the subject demanding access, the action the subject wants to perform, the resource being accessed and the environment or context in which access is requested.

In ABAC permissions are defined in terms of privilege-giving attributes. Instead of defining new roles to represent sets of access permissions ABAC defines the permission sets by combining the privilege-giving attributes. For example, these three attributes being an employee, having a driving license and being a Swedish citizen may in different combinations give different sets of access permissions. Potentially there would be $8 \times 2^3 = 8$ different sets of access permissions, hence in RBAC 8 different roles that can be named and need to be managed. But in ABAC only the three well-understood privilege-giving attributes need to be managed, which is of course a much simpler task than managing 8 different roles that in most cases do not have any meaning in the organisation.

Here’s another example that shows the power of ABAC compared to RBAC: A university account manager might be authorised to make bank transfers from an application. In RBAC, this would usually mean that he would have a role enabled for his account, for example “Make_Transfers”. As the need for control gets tighter, the manager may be authorised only to make transfers up to a value of 2000 EUR without any approval. Anything else above that requires the approval of at least two of the financial supervisors. Implementing this with RBAC is not easy. The bank transfer application would have to have some hardwired piece of logic implementing the “max 2000 EUR without approval”. With ABAC, the policy could just express that if the account manager has the role “Make_Transfers” and attribute “transfer_amount <= 2000” the operation is approved. Also approved is an operation if someone has the role “Make_Transfers” and attributes “transfer_amount > 2000” and “valid_approvals >= 2”. Everything else is denied.

This all-inclusive approach makes ABAC suitable for situations in which finer granularity and context-aware authorisations are required.
Besides allowing for fine-grained access control, ABAC has more advantages that make it suitable for utilisation in large-scale distributed systems such as federations. These advantages are:

- It allows for personalisation based on the provided attributes.
- It facilitates automatic provisioning of user identities.
- It takes away the need for having an account at several SPs.
- Attributes are always up-to-date.
- It allows for better implementation of license conditions.

The use of attributes for authorisation, however, comes with several challenges such as the trustworthiness of the attributes, their semantic interpretation, and the definition of a syntax for expressing attribute-based authorisation queries and responses (see section 5). In other words, ABAC offers flexibility but introduces complexity.

### 2.2.2 Authorisation models

#### 2.2.2.1 Entities

Looking at the authorisation use-cases, at a very high level the following three entities are part of the authorisation data model:

1. **Subject** – an entity that can request, receive, own, transfer, present or delegate an authorisation to excise a certain right. Informally, a subject is any user of a service or resource. Subjects are organised into groups, roles, institutions, federations, etc. Subjects take action to use a resource and have attributes.

2. **Resource** – a component of the system that provides or hosts services and may enforce access to these services based on a set of rules and policies defined by entities that are authoritative for these particular resources. Resources may be organised into hierarchies, groups, etc. and may be described in terms of attributes. Policies bring together the subject, resource, actions, their organisations and attributes.

3. **Authority** – an administrative entity that is capable of and authoritative for issuing, validating and revoking a means of proof such that the subject is authorised a certain right.

There are three general types of authorities in common use. Attribute authorities, which issue attribute assertions that relate to the subject. Policy authorities, which issue authorisation policies with respect to resources and services offered by these resources. These authorisation policies contain assertions that a given subject has a certain right with respect to a given service. Identity authorities (e.g. the CA in a PKI or the IdP) that issue assertions related to the identity of the subject and are typically based on the credentials that are used during an authentication process.

#### 2.2.2.2 Sequences

Given this set of entities, several authorisation sequences can be defined:

1. **Push sequence** – the subject first requests an authorisation from an authority.
2. **Pull sequence** – the subject will contact the resource directly with a request upon which the resource contacts the authority.
3. Agent sequence – the subject will contact a higher level agent with a request to obtain a service authorisation. This agent will make an authorisation decision based on the rules established by the authority and if successful it will contact the resource.

The three sequence models are shown in Figure 3.

![Figure 3: Push (left), pull (middle) and agent (right) authorisation sequences.](image)

Pull sequences are commonly used in the network world with systems using the RSVP or RADIUS protocol where requests typically are carried “in-band”. Examples of push sequences are found in many ticketing systems such as Kerberos. In identity federations, the push sequence is also favoured. Here, the identity provider (IdP), provides subject identity assertions to SP. Sometimes, the SP may request for additional attributes. In that case, a pull mechanism is used. The agent sequence is relevant to Grid users when requesting a certain QoS from the Grid system (e.g., resource reservation through a scheduler). A similar model has been proposed for light-path provisioning as well [17].

The agent model constitutes a decentralised approach to access control policy management. Policies need to be expressed, managed and enforced in distributed agents that are located at the perimeter of every domain where services are enforced. A single administrative domain may have multiple sub-domains where services are located and may require multiple agents to control access to those services. In case of push and pull models, policies can be managed centrally and applied to a wide group of services located in different domains. Therefore, both models fit well into web services based architectures as they provide means of centralising authorisation information in specialised servers [18].

2.2.2.3 Tokens

In the ABAC model attributes are often packaged as a security token (i.e. claim, ticket or assertion). The token is issued by an authority (IdP or Security Token Service (STS)) and cryptographically protected.

The University of Amsterdam extensively studied the role of tokens for communicating authorisation decisions and security context between domains [19].
To allow multi-domain resource access, the domains must interact in a coordinated manner. Two typical coordination approaches can be distinguished: The chain and tree approach. The chain approach is typical for multi-domain network provisioning scenarios used amongst Network Service Providers. The tree approach is typically used for Grid scenarios.

In the chain approach the token exchange may be as follows: When a user requests an authorisation from an authority to use a service in a multi-domain environment, each domain’s authority will apply some policy when evaluating a request. Policies may imply rules and/or conditions regarding the identity of the requestor and its authorisations. Each domain may have its own policy what will imply a specific domain related context to a decision that the token will represent. Figure 4 illustrates interactions between major entities participating in a multi-domain service provisioning chain approach scenario.

In Grid environments, the network resource may be provisioned in the same way as any other Grid resource. Grid applications typically use a centralised entity as common authority for this purpose. Figure 5 shows such a centralised or tree approach.
Within the tree approach, a common authority will negotiate with the individual domain authorities. If the common authority can resolve the request, it will provide a token to the user to indicate all involved domains are committed to provide the requested resource. Alternatively, each domain can create a token and associated privileges, where the common authority just passes it on to the user. In this case, the user needs to insert a number of tokens into the response, one for each domain.

2.2.2.4 Functional architecture

Authorisation information such as policies, attributes, identities and context parameters (e.g. transaction-related, time, location) are utilised and combined when making authorisation decisions. An authorisation architecture consists of a set of entities and functional components that allow for these decisions to be made and enforced. An example authorisation architecture is depicted in Figure 6.

![Functional authorisation architecture](image)

Figure 6: Functional authorisation architecture (based on the pull sequence).

Key functional components in this architecture are:

- Identity and group management for the provisioning of identity authentication credentials and attributes that may amongst others express group contexts.
- Policy management for the provisioning of policies, their consistency (configuration management), and related metadata.
- Policy-based authorisation for the actual mediation of access to a federated service or resource via the Policy Decision Point (PDP) and the Policy Enforcement Point (PEP). The PDP makes authorisation decisions about a subject’s access to a service or resource based upon the available policies and provided identity assertions; the PEP enforces the actual access.

Sometimes the environmental or situational context such as time or location is also taken into account during the authorisation process. Naturally, all authorisation-related activities are audited and logged.
Modern authorisation architectures encapsulate all policy evaluation into a PDP, which is separated from the PEP. This has several advantages: it simplifies changing policies across many systems, it makes the policy much easier to analyse, and it avoids the risks of incorrectly implemented policy logic scattered throughout application code. In order for this model to be effective, it must be followed throughout the whole infrastructure. The architecture may contain multiple PEP and PDPs that are distributed across multiple domains.

2.2.2.5 **Process-view**

The authorisation process consists of several basic security services that operate between the subject’s request for resource access and the actual denial or granting of the requested access. These services provide for authentication, attribute collection, policy retrieval, policy decision and enforcement. The whole authorisation process is depicted in Figure 7.

![Diagram](image)

**Figure 7: Authorisation process (from [20]).**

Instead of letting the resource or application take care of the whole process itself, several of these services can be externalised. In most of today’s identity federations authentication has already been externalised and is provided by the subject’s home IdP. Externalising authorisation is barely seen but might be the next step towards a more flexible and scalable access control infrastructure that suits the federated identity model.
2.2.3 Authorisation mechanisms

2.2.3.1 Externalising authorisation

At the onset, authorisation appears to be an application-specific problem. After all, it is the application and its resources that need to be protected. Traditional applications often define proprietary policies that can only be enforced and administered by proprietary mechanisms. With applications and users scattered across multiple SPs and IdPs in a federation, administration of these proprietary authorisation policies becomes a nightmare. There is a need for a more centralised authorisation policy management. This can be achieved by externalising authorisation intelligence from the application and let the application only take care of enforcing the authorisation check (see Figure 8).

Externalised authorisation separates authorisation logic from business logic, resulting in increased transparency as the entitlements are now understandable and measurable beyond the context of the application. This provides a service provider a consistent view of his authorisation policies spanning multiple services.

One of the main benefits of externalising authorisation is centralisation of policy management. In case of a changing business rule, there is no need to change all the local application policies. Only the central policy has to be updated. Considerations that drive a centralised and shared authorisation service are [21]:

- Separation of business logic from authorisation. In many applications business logic can be separated from authorisation which is an orthogonal functionality. Authorisation can be managed independently and plugged easily into composite distributed applications without the need of prior knowledge about those applications.
- Separation of duties. Authorisation policies can be written independently by security specialists, service owners or administrators and do not need to be coded at the same time and in the same package as business services. This facilitates audits and checks of security policies for the purposes of correctness, governance and compliance.
• Enhanced authorisation decisions. When authorisation is offloaded to specialised components then possibly more accurate authorisation decisions can be made as well. This is because authorisation services often have a more detailed view of the entire computing environment and may introduce useful information into their policy evaluation process. Additionally, authorisation services easily contribute to uniformity of accounting and auditing functions as discussed in [22].

• Consolidation of policies: centrally managing policies means consistent access and role mapping decisions across all applications.

• Scalability: separating concerns between PEP and PDP effectively scales the authorisation infrastructure. When the PDP operates as a standalone service it can handle decision requests from multiple PEPs.

• Improved policy lifecycle management: Authorisation policies can evolve very differently from application requirements. Having the authorisation logic hard coded into the business logic means changing code each time there is a policy change. By keeping authorisation policies outside of the application, they can be changed without modifying the application.

• Better policy overview and control: Policies controlling actions on several depending services can be made visible.

• Delegation: delegation of authorisation is a common and difficult problem for any developer. An account manager might delegate authorisation to his manager while he is on vacation. The developer has to take care of this delegation appropriately across multiple applications. A single policy interface will simplify his job.

• Audit: If each application has its own way of defining policies, an auditor may have to deal with inconsistent formats spanning multiple applications of various flavours in order to obtain the full picture of user entitlements. A centralised policy makes auditing a lot easier.

There are also drawbacks related to externalising authorisation:

• Complexity: the centralised policy may become very complex as it may contain policies for numerous applications.

• Administration: delegated administration will be required as application controllers may know best how to specify the policy for their application. This will become even more difficult to handle across multiple organisations (see federated authorisation below).

• Performance: the central PDP may become a bottleneck in all authorisation traffic. How readily available are the policies and how much external information does the PDP need to fetch to evaluate a policy? SLAs for decision latency might be required and easily violated.

Externalised authorisation has been known for many years now and has been first discussed by Woo and Lam [22]. In their model, services simply need to enforce authorisation but delegate the decision making process to other specialised components.

To efficiently support authorisation externalisation, the following services must be present as part of the authorisation infrastructure to provide the needed functionality:
- A Policy Administration Point (PAP) where administrators can create and modify application specific policies.
- A Policy Enforcement Point (PEP) that is responsible for triggering authorisation policy decisions within applications. The PEP could be part of an application but may also be shared among applications. In the latter case the PEP is external to the applications.
- A Policy Decision Point (PDP) that provides the actual authorisation decisions on behalf of the PEP.
- One or more Policy Information Points (PIPs) which supply data and information such as subject or resource attributes to the PDP to help it make accurate policy decisions.

The centrepiece of this infrastructure is the PDP. It must be able to take requests from the PEP, such as “Can this user use this resource?”, and find the appropriate policies in order to come to a decision. The PDP will consult its PIP(s) and return back a Grant or Deny response in addition to any optional obligations that the decision entails. An obligation may be to log the user’s activities once access has been granted.

The centralised policy administration is provided through the PAP. Authorisation policies are defined and consolidated into a single policy store across applications. The PAP provides a consistent user experience for policy management. It also ensures a consistent format in how authorisation policies are represented. From a compliance standpoint, operations such as audit and attestation will both benefit from the centralisation of user authorisation. In a federation, centralised administration through the PAP can only be achieved if PAP authorisation is delegated to policy managers of other federation members, to allow them to set the proper policies for their users. This is for instance supported by the XACML policy language (see section 3.1.1).

2.2.3.2 Federated authorisation

Traditionally, the PDP, PEP, PIP and PAP reside at the service provider side. In this model, a given user has a profile consisting of a set of attributes stored at an attribute provider (which may or may not be the same as that user’s identity provider). The user goes to the service provider and, via a redirect to the user’s identity provider, the service provider receives an assertion containing the required set of attributes for controlling access. These attributes are specified at the PAP and obtained via the PIP. The service provider acting as the PDP decides on the basis of the user’s identity (including the attributes), which resources the user may access, and acting as the PEP, restricts the user’s access appropriately. This model is most common in today’s federated authorisation infrastructures and is shown in Figure 9.
In another approach, the service provider is still a PEP, but the PDP is elsewhere. The PDP could be implemented by the identity provider. In this case, the service provider requests an authorisation decision from the identity provider, without being party to the data underlying the decision. This second model can be called *federated authorisation*, i.e. it is the identity provider that decides what the user is allowed to do at the service provider side and not the service provider. The service provider not only trusts the authentication assertions of the identity provider but the authorisation assertions as well. Figure 10 depicts the federated authorisation model. In this figure the PAP and PIP are located at the identity provider as well. The PAP basically serves as a policy repository for the PDP and the PIP knows where to obtain the attributes.

Looking at the federation protocols, both SAML and WS-Federation enable the first model – passing attributes in assertions that are themselves carried in authentication responses. XACML is the basis for the second model, and is profiled for use in SAML by the SAML2.0 profile of XACML2.0 [23].
Other distribution models can be imagined as well. The PAP could be hosted by the IdP or by the SP. What matters here is that each party is empowered and facilitated to manage its own policies. The delegation profile of XACML3.0 facilitates decentralised administration of authorisation policies, letting SPs, IdPs and even users define and manage policies in their own domains.

In a hub and spoke federation even more PDP/PEP/PIP/PAP configuration variants are possible. The hub is trusted by all SPs and IdPs in the federation and as a central and coordinating entity could certainly play the role of PIP, as well as PDP, PAP and PEP. Important to take into account is that the party that is hosting the PIP must be authorised to collect the attributes that are needed for the PDP to make the right policy decisions. The hub typically is such an authorised and coordinating entity and could very well host the PIP. Also having PDP functionality at the hub would reduce the communication overhead between the SP and the hub, i.e. the PIP and the PDP can communicate locally with each other. The location of PAP functionality is less relevant as its administration can be delegated. Figure 11 shows several realistic PAP/PDP/PEP/PIP combinations in a hub-based federation.

Figure 11: Possible PAP/PDP/PEP/PIP combinations in a hub-based federation.

Despite the advantages of externalising authorisation, little uptake of the federated authorisation model is observed in today’s federations. One may wonder if federated authorisation is a valid model. Even in the SURFfederatie model, where a student at university A is granted access to a resource at university B, the authorisation decision is made by university B (yes, it’s based upon the federated data that the student is a student at university A, but the authorisation decision is made by university B).
An obvious reason for this lack of uptake is that XACML is not commonly used by identity and service providers; they mostly rely on SAML for communicating authorisation or attribute assertions. There is resemblance with the introduction of SAML in many federations. Here also not all identity and service providers were capable of talking SAML or spoke another language (e.g. A-Select or WS-Federation). This resulted in the development of a hub in many federations including the SURFfederatie. The hub is able to perform protocol transformations, allowing federation members to talk to each other. Introducing XACML in a federation may require a similar hub-based approach. The hub could e.g. facilitate SAML-to-XACML authorisation transformations and adopt a model that is best described as ‘authorisation as a service’, i.e. the hub provides authorisation services for its federation members.

Referring to several articles/blog posts on the topic\textsuperscript{1,2,3,4}, these all bring out good examples of some form of federated (or delegated) authorisation. Though these use cases are interesting, the federated authorisation model has some issues that prevent its widespread adoptions including:

- The range of settings for permissions at any one resource controller are vastly different from one instance of that type of resource to another, thus substantially raising the complexity of any centralised management infrastructure. For example, in a typical "authorise my lawyer to pay my insurance bill" delegation scenario, he not only has to be authorised access to the bank account, but to extract money for a particular purpose and probably with some limits on the amount as well.
- The complexity for a good user understandable interface for interacting with federated authorisation and delegation settings. A lot of work still needs to be done in the realm of computer/human interfaces to provide a generic mechanism that will allow a central server to have the knowledge and understanding to walk the average technologist through the process, let alone an average user.
- There are a lot of privacy considerations around a centralised authorisation entity (much more so than a centralised identity entity). This one entity will not only control all of the user’s authorisations, but will also have the knowledge of what services a user has been granted access.
- Doing authorisation remotely can have a significant negative performance impact. This comes from the messages necessary to obtain the authorisation information, the caching of said information and the parsing of it on each access. An internal authorisation solution can be optimised for the resources being accessed and even tied to the resources within the internal database of the application. Such tight coupling is very hard to do, if not impossible, when receiving authorisation statements from remote parties.

\textsuperscript{1} Pat Patterson’s Superpatterns blog: Federated Authorization, see \url{http://blogs.sun.com/superpat/entry/federated_authorization}.
\textsuperscript{2} Conor’s Web Log of Esoterica: Federated Authorization, see \url{http://conorcahill.blogspot.com/2006/12/federated-authorization.html} and \url{http://conorcahill.blogspot.com/2006/12/federated-authorization-3.html}.
\textsuperscript{3} James McGovern’s blog on Enterprise Architecture: From Incite comes Insight...: Consumer Perspectives on Federated Authorization, see \url{http://duckdown.blogspot.com/2006/12/consumer-perspectives-on-federated.html} and \url{http://duckdown.blogspot.com/2006/12/even-more-thoughts-on-federated.html}.
\textsuperscript{4} Shekhar Jha’s blog on Identity and Access Management: Federated Authorization and Relationship, see \url{http://identityaccessmanagement.blogspot.com/2006/12/federated-authorization-and.html}.
It seems that tight, application specific authorisations (such as "can James add a comment to this article on my blog") require the more traditional approach where decisions are kept with the resources being authorised. Loose, granular, cross-application authorisations (such as "can James see my blog") are more suitable for federated authorisation.

Maybe a hybrid of the authorisation push sequence (see section 2.2.2.2) and local policy evaluation is more appropriate for the federation model where subject authorisations will be sent to the other domain alongside its identity. This approach is for instance defined by SecPAL (see section 3.3.4 and [24]) and commonly used in modern SAML-based federations (see 3.1.2).

## 2.3 Summary

Policies show great potential as a way to control the behaviour of complex computer systems. In the case of authorisation decisions in large distributed and federated systems, policies offer the potential to abstract away from the details of who is allowed to access which services, under which conditions. This layer of abstraction is both a challenge and an opportunity: policy-driven distributed authorisation systems may be more manageable, scalable, available, and secure than previous approaches or they may be just the opposite.

Traditional authorisation models like RBAC or DAC cannot cope with the diversity of resources in identity federations. ABAC, however, offers more flexibility and can handle heterogeneous resources and user groups. The flexibility afforded by attribute-based policy systems, however, comes at the cost of increased system complexity.

The value proposition of externalising identity is starting to increase. While SAML2.0 looks to be the de-facto solution for solving federated authentication, there is still debate as to what mechanisms and protocols to use to address how authorisation decisions are made in federations. Service providers have not yet taken the next step to externalise authorisation and to use approaches based on for instance XACML. The reason for this might be due to the fact that, compared to authentication, authorisation is much more application specific and complex.

What the user can do in one application often varies significantly from what he can do in another. Especially in commercial-off-the-shelf applications, authorisation is generally more naturally handled by the application. However, there may be types of applications that do not vary in their authorisations. For these types of applications externalising and federating authorisation might be worth while. Performance is then a concern that should be taken into account. The incurred network costs for remote authorisation can by far exceed the actual cost of a local authorisation decision.

The next section describes the state-of-the-art of policy-based authorisation solutions.
3 Policy-based authorisation solutions

Policy-based authorisation solutions can roughly be grouped into four distinct clusters: web-based, user-based, Grid-based, and privilege management infrastructure based (PMIs). The solutions in the first group are SAML, XACML, and WS*. User-based solutions are User Managed Access (UMA) and OAuth. Grid solutions consist of OGF, Globus and Akenti, while PERMIS and Signet make up the PMI group.

This chapter will describe the solutions for each cluster in more detail.

3.1 Web-based

This section lists the web-based solutions for policy-based authorisation.

3.1.1 XACML

3.1.1.1 Overview

A relatively new technology to express access control policies in the web services arena is the eXtensible Access Control Markup Language (XACML, [25]).

XACML is an OASIS standard that aims to specify the following:
1. A general-purpose access control policy language;
2. An access control decision request/response protocol.

The access control policy language provides syntax in XML for defining action (type of request) rules for subjects (users) and targets (resources). The authorisation request/response protocol defines a format of messages and the information flow between enforcement and decision components of the authorisation system. It is based on the Attribute Based Access Control (ABAC) model. Attributes that are exchanged between components of the system can be encoded using the Security Assertion Mark-up Language (SAML). SAML allows exchanging standard authorisation data between components and is suitable for use in conjunction with XACML.

XACML supports interoperability between domains of trust so that separate components of the authorisation architecture can work together across domains. It aims to replace proprietary policy languages or formats that apply to specific applications only. This enables a consolidated policy view across the entire computing environment, i.e. federation. Defining a standard XML-based syntax also aims to address management related issues. It eases development of standard management tools and toolkits that would serve common policy needs. Those tools can be deployed in a centralised manner to reduce operational costs [26].
The SAML profile for XACML defines how to use SAML to protect, transport, and request XACML messages and other information in XACML-based authorisation systems [23].

The XACML data flow diagram along with the enforcement and decision components is depicted in Figure 12.

![Figure 12: XACML data-flow diagram.](image)

The data flow model of XACML follows the pull model of the authorisation decision query sequence as is presented in Figure 4 of section 2.2.2. Briefly, the access request is received by the policy enforcement point (PEP) which then communicates it to the decision point (PDP). The PDP evaluates the access request with regards to the applicable policy set, policy or rule and replies with an authorisation decision. In order to make a decision, the PDP obtains attributes associated with the client issuing the access request, the resource that is being accessed and the environment in which the access request is taking place. Such attributes are retrieved from the policy information points (PIPs). The full message sequence description is explained in the specification [25].

Decisions, made by PDP, do not only specify whether access should be granted or not but may additionally impose certain obligations on enforcement points. Those obligations are an important feature of the authorisation system [27]. They allow defining actions, which the PEP must perform prior to giving access to the client. The use of obligations minimises the set of policies that needs to be composed. XACML does not specify how policy obligations should be defined within authorisation decision messages. Therefore, a bilateral agreement between the components of the authorisation system must exist, as PEPs need to understand obligations defined within policies stored by administration points (PAPs).
The first version of XACML was published in 2003, and since then there has been ongoing work on new versions – currently, XACML3.0 is close to standardisation. The implementation of delegation is new in XACML3.0. The delegation mechanism is used to support decentralised administration of access policies. It allows an authority (delegator) to delegate all or parts of its own authority or someone else’s authority to another user (delegate) without any need to involve modification of the root policy. In this delegation model, the delegation rights are separated from the access rights. These are instead referred to as administrative control policies. Using these delegation features, the broad authorisation policies for group communication can be specified by a coordination authority, while power to create fine-grain authorisations (e.g. for specific subjects) can be delegated to the organisations participating in the group.

The power of XACML lies in the fact that access control is no longer something that resides inside the application, but is externally managed using a standardised policy language. Therefore, different applications can use the same policies, and these policies can be hierarchically defined at the federation level, the institution level, or the application level.

XACML has many benefits over other access control policy languages:
- One standard access control policy language can replace dozens of application-specific languages
- Administrators save time because they don’t need to rewrite their policies in many different languages
- Developers save time because they don't have to invent new policy languages and write code to support them. They can reuse existing code.
- XACML is flexible enough to accommodate most access control policy needs and extensible so that new requirements can be supported.
- One XACML policy can cover many resources. This helps avoid inconsistent policies on different resources.
- It’s distributed. This means that a policy can be written which in turn refers to other policies kept in arbitrary locations, i.e. XACML allows one policy to refer to another. The result is that rather than having to manage a single monolithic policy, different people or groups can manage separate sub-policies as appropriate, and XACML knows how to correctly combine the results from these different policies into one decision. This is important for federations. For instance, a site-specific policy may refer to a federation-wide policy and a service provider specific policy.

Several XACML drawbacks are:
- Its complexity and consequently steep learning curve required by administrators to write policies.
- Good tools for writing and managing XACML policies in an intuitive and user-friendly manner are lacking.
- Poor performance characteristics (see section 5.1.3).
- Service customisation is required in order to make them XACML-enabled.
3.1.1.2 In federations

XACML is one of the approaches that provides an interoperable solution for authorisation in identity federations. This can be considered from different perspectives including its:

- **Modularity of the system and its ability to span different administrative domains.** Multiple PEPs can use different PDPs of their choice. Those PDPs may be located in separate administrative domains and use arbitrarily chosen information and administration points for attribute and policy retrieval.
- **Ability to compose policies or policy sets from distributed sources.** In XACML, policies can be composed of a variety of distributed policies and rules that can be managed by different organisational units. Decisions can be derived from multiple rules (rule combining algorithms) or multiple policies (policy combining algorithm). In both cases it is up to the systems administrator to define which combining algorithms should be used (e.g. first applicable, deny overrides, etc.).
- **Means to provide policy administration in the federation.** This includes the XACML Administration and Delegation profile [28] which extends policy schema to describe delegation policies. This profile also extends the request context schema to describe administrative requests.

3.1.1.3 Interoperability and profiles

Back in February 2007, Burton Group issued a challenge to the industry to demonstrate interoperability of XACML. The industry responded, via the OASIS XACML TC, in June 2007 by having the first XACML Interoperability Demo at the Burton Group Catalyst conference. There were two particular use cases in this demo, which required interoperability between vendor implementations of PEPs, PDPs and PAPs in terms of authorisation requests/responses and policy exchange. The former term is most interesting for this document, and during the interoperability event some specific choices were made in order to make this work:

- **Implementation of the XACML PDP interface as a SOAP interface which accepts a XACMLAuthsDecisionQuery and returns a XACMLAuthsDecisionStatement which are contained in the SOAP body.**
- **Use of the SAML 2.0 Profile for XACML 2.0 which defines a Request/Response mechanism for carrying xacml-context:Request and xacml-context:Response elements.**

While many vendors claimed conformance and implementation of the XACML 2.0 standard after the interoperability event, their PDP interfaces are still proprietary and unique. These interfaces may be implemented using web services but each web service implementation (e.g. WS-Security) is unique and special to that vendor and does not follow any consistent interface specification and as such is an integration exercise that is left up to implementers if PEPs from multiple vendors need to talk to a XACML PDP [29]. To reduce this integration effort, the OpenAz project has been initiated (see section 5.1.1.1).

The SAML2.0 profile for XACML2.0 has already been mentioned as an appropriate back channel protocol to convey the XACML authorisation queries and responses between SPs and IdPs. Additional profiles are under development:
• The Cross-enterprise Security and Privacy Authorisation (XSPA) profile of XACML describes mechanisms for authenticating, administering, and enforcing authorisation policies that control access to protected information residing within or across enterprise boundaries in the health domain. The profile promotes interoperability within the healthcare community by providing common semantics and vocabularies for policy enforcement [30].

• The web service profile of XACML (WS-XACML) specifies ways to use XACML in the context of web services for authorisation, access control, and privacy policies. It specifies four types of information.
  1. An authorisation token or credential based on XACML to be used in a web services context for conveying an authorisation decision from a trusted 3rd party to a Web Service.
  2. A policy assertion type based on XACML elements for use with WS-Policy or other schemas and protocols. The profile specifies standard formats, matching semantics, and usage guidelines for two assertions derived from this type: one for authorisation policies and the other for privacy policies.
  3. Some ways in which attributes for a client may be passed to a web service as part of a SOAP message in such a way that they can be authenticated as having been issued by a trusted authority. These attributes may be used by the web service in evaluating the internal XACML policies of a service that are relevant to a given web service’s access.
  4. How to express privacy preferences and match them using the new assertion based on XACML. Unique for WS-XACML is that it facilitates initial policy matching of the server’s requirements with the client capabilities and vice versa.

In WS-XACML, both parties may state a policy assertion. Each party may have Capabilities and Requirements. Capabilities are information that the party is able and willing to provide in conjunction with having its Requirements met. Negotiation protocols have not yet been discussed. Each party might make its Requirements known to the other party as a first step. The other party then determines whether its Capabilities can satisfy the Requirements, and if so, the minimal set. Each party might then send its minimal set of Capabilities to the other party.

So far, WS-XACML is defined only for determining policy compatibility, not for negotiating binding agreements. Providing a copy of the XACML assertion with minimal Capabilities to the other party in a signed SAML assertion with a validity period, however, might be a format for use in a binding agreement.

3.1.2 SAML-based authorisation

An alternative approach is to push SAML attribute assertions during the authentication. Those attributes are then used for authorisation decisions by the service. In a lot of federations this use of SAML is common practice and may make XACML a hard sell. Moreover, Microsoft supports SAML tokens in ADFS V2; they don’t support XACML.
But things are more complicated. If all that is needed is to pass attributes into the application, then SAML will be sufficient. If more complex manners of expressing and storing policies about the content of those attributes and the resultant set of decisions should be based on the expression of those policies, then more is required. That, and more, is precisely what XACML is.

3.1.3 **WS-***

3.1.3.1 **WS-Policy**

The Web Services Policy Framework (WS-Policy) provides a general purpose model and corresponding syntax to describe the policies of a web service. WS-Policy defines a base set of constructs that can be used and extended by other Web services specifications to describe a broad range of service requirements and capabilities.

WS-Policy can express requirements, capabilities and assertions of a particular web service [31]. For example, a policy can indicate that a web service only accepts requests containing a valid signature or a certain message size should not be exceeded. How a policy can be obtained is out of the scope of the specification.

WS-PolicyAttachments defines several methods for associating the WS-Policy expressions with web services (i.e., WSDL).

A comparison between XACML and WS-Policy is made in [32]. Although WS-Policy and XACML overlap in some areas of functionality, there are actually large inherent differences between the two specifications. WS-Policy is a more general policy language, while XACML is by nature an access control rule language.

3.1.3.2 **WS-SecurityPolicy**

While WS-Policy defines a framework for allowing web services to express their constraints and requirements in terms of policy assertions, WS-SecurityPolicy defines a set of security policy assertions for use with the WS-Policy framework with respect to security features provided in WS-Security: SOAP Message Security, WS-Trust and WS-SecureConversation.

The assertions defined within the WS-SecurityPolicy specification have been designed to work independently of a specific version of WS-Policy. WS-SecurityPolicy takes the approach of defining a base set of assertions that describe how messages are to be secured. Flexibility with respect to token types, cryptographic algorithms and mechanisms used, including using transport level security is part of the design and allows for evolution over time. The intent is to provide enough information for compatibility and interoperability to be determined by web service participants along with all information necessary to actually enable a participant to engage in a secure exchange of messages.
3.2 User controlled

The rapidly developing web environment provides users with a wide set of rich services as varied and complex as desktop applications. Those services are collectively referred to as “Web 2.0”, with examples such as Google Docs, Wikipedia, Wordpress, Youtube, or Flickr, that allow users to create, manage and share their content online. By switching from desktop applications to their Web equivalents more and more data gets released online. It is the user who creates data, who shares and disseminates this data, and who accesses it.

Storing and sharing resources over a highly collaborative “Web 2.0” environment poses new security challenges. Access control, in particular, is currently poorly addressed in such an environment and is not well suited to the increasing amount of resources that are available online. This has given rise to several user-controlled access solutions.

3.2.1 OAuth

OAuth’s stated objective is to create an ‘authorisation delegation protocol’ [33]. In practice, OAuth is an open protocol built for web-based applications that allows users to share their private resources (e.g. photos, videos, contact lists) stored on one site with another site without having to hand out their username and password.

OAuth allows users to hand out tokens instead of usernames and passwords to their data hosted by a given SP [34]. Each token grants access to a specific site (e.g. a video editing site) for specific resources (e.g. just videos from a specific album) and for a defined duration (e.g. the next 2 hours).

Thus OAuth allows a user to grant a third party site access to their information stored with another SP, without sharing their access permissions or the full extent of their data.

OAuth authorisation for web applications involves a sequence of interactions between a web application, services, and the end user: For Google services the sequence is as follows:

1. The web application contacts the Google Authorisation service, asking for a request token for one or more Google service.
2. Google responds with an unauthorised request token.
3. The web application directs the end user to a Google authorisation page, referencing the request token.
4. On the Google authorisation page, the user is prompted to log into their account (for verification) and then either grant or deny limited access to their Google service data by the web application.
5. The user decides whether to grant or deny access to the web application. If the user denies access, they are directed to a Google page and not back to the web application.
6. If the user grants access, the Authorisation service redirects the user back to a page designated with the web application that was registered with Google. The redirect includes the now authorised request token.
7. The web application sends a request to the Google Authorisation service to exchange the authorised request token for an access token.
8. Google verifies the request and returns a valid access token.
9. The web application sends a request to the Google service in question. The request is signed and includes the access token.
10. If the Google service recognises the token, it supplies the requested data.

The sequence is shown in Figure 13.

Figure 13: OAuth message sequence.

OAuth is primarily designed to allow 3rd party applications to do actions on behalf of authenticated users (3-legged OAuth). There is also a 2-legged OAuth variant that allows applications to sign requests. This allows services that receive a request to validate that those requests came from the application. 2-legged OAuth is applicable in closed domains in which the services trust each other. It basically involves only steps 9 and 10 of Figure 13 and does not require any user interaction.

Improvements of OAuth are specified under the name WRAP (Web Resource Authorisation Profiles [35]). WRAP attempts to simplify the OAuth protocol, primarily by dropping the signatures, and replacing them with a requirement to acquire short lived tokens over SSL. WRAP addresses two areas in which the original OAuth (version 1.0a) protocol is lacking: it offers new ways to obtain tokens, and it evolves the architecture to enable other roles to issue tokens (other than the server).
OAuth 1.0a offers a single browser-based redirection flow used to send the user from the application to the server, obtain approval, and return to the application. WRAP adds a few new flows for obtaining authorisation and tokens mainly designed around providing better experiences on devices, desktop applications, or fully JavaScript based implementations. Unlike 1.0a where the server issues and verifies every token, the tokens in OAuth WRAP are short lived and can represent claims issued by an authorisation server, providing scale and security benefits for large operators. Note that WRAP uses HTTPS only for obtaining tokens but does not mandate (or even suggests) using HTTPS for making protected resources requests. Instead, WRAP recommends short lived tokens that must be refreshed (using HTTPS). In other words, WRAP uses secrets that are easy to steal, but which are good for a short period of time, limiting their damage.

Just recently, the IETF OAuth Working Group published the first draft of the OAuth2.0 protocol [36]. OAuth2.0 is a completely new protocol and is not backwards compatible with previous versions. However, it retains the overall architecture and approach established by the previous versions. The following is a subset of the new features available in OAuth 2.0:

- New communication flows for user agents, web servers, devices, and assertions (i.e. the client presents an assertion such as a SAML assertion to the authorisation server in exchange for an access token).
- A cryptography-free option for authentication which is based on existing cookie authentication architectures. Instead of sending signed requests the token itself is used as a secret sent over HTTPS.
- Signature support has been significantly simplified to remove the need for special parsing, encoding, and sorting of parameters. It also uses a single secret instead of two.
- Instead of issuing a long lasting token (typically good for a year or unlimited lifetime), the server can issue a short-lived access token and a long lived refresh token. This allows clients to obtain a new access token without having to involve the user again, but keeps access tokens limited. This resembles the short lived proxy credentials that are used as access tokens in the Grid community.
- OAuth 2.0 separates the role of the authorisation server responsible for obtaining user authorisation and issuing tokens from that of the resource server handling API calls.

Though OAuth2.0 provides a basic assertion flow, the flow itself is somewhat useless without further profiling, or at least a private understanding between the various parties in the OAuth exchange. It is clear that a profile of the assertion flow for use with SAML is required in order to provide for vendor/software interoperability. For this reason a SAML Assertion Flow Profile is proposed that builds upon the OAuth2.0 Assertion Flow by specifying the exact format and assertion values to be used with SAML2.0 [37]. Specifically, it is intended to provide interoperability with the SAML2.0 Web Browser SSO Profile and the SAML HTTP POST Binding by allowing the authorisation server to process SAML assertions with the same format and characteristics as those used in SAML Web SSO.
3.2.2 User-Managed Access

UMA is a user centric and OAuth-based web protocol that lets the user control authorisation of data sharing and service access made between online services on his/her behalf. The UMA protocol is currently being developed in the Kantara UMA Work Group [38]. Version 1.0 of the UMA core protocol has been recently specified [39].

UMA provides a dedicated interface and service for:
- Authorising data sharing and service access;
- Imposing sharing terms on any application wanting access;
- Monitoring, changing, and stopping access relationships;
- Letting services make requests of all of your authoritative sources directly.

The following scenario illustrates the working of UMA:
1. Alice chooses to store her research data at her university.
2. She introduces the university to her chosen Authorisation Manager (AM) for data protection.
3. She sets up policies and terms for that data at the AM.
4. She tells Bob, a researcher of another university, where to find her data.

---

5 XRD = eXtensible Resource Descriptor; LRDD = Link-based Resource Descriptor Discovery; see http://hueniverse.com/2009/03/the-discovery-protocol-stack/ for more information about XRD and LRDD.
5. Bob tries to retrieve the data and gets introduced to the AM as a result.

6. Bob and the AM negotiate terms for access.
7. Bob retrieves the data, once the AM tells the university it’s okay to release it (as often as Alice’s policy allows it).

8. One day, Alice revokes access, blocking Bob’s data access.

A more detailed scenario can be found at the Kantara site of the working group [40, 41].

### 3.2.3 Lockr

Lockr is an access control system based on social relationships that separate content delivery and sharing from managing social networking information [42, 43]. Users are able to maintain a single social network and base Access Control Lists for different applications on social relationships.

To illustrate how Lockr works, consider a simple example of a person wanting to restrict access to their family photos on Flickr. The owner creates a social ACL indicating that access to the photos is restricted to family-only. Family members must present their social attestations to Flickr issued by the photos’ owner before gaining access. To allow access, Flickr must verify that the attestations were issued by the original owner and that “family member” is the social relationship encapsulated by the attestation. Note that the family members’ social attestations can be reused by any online site without requiring users to register.
Users need to manage a single social network that can be stored in an address book on their own machines. To create an access control policy, users do not have to enumerate all members of a social group. Instead, they can list what social relationships others must have to gain access to personal content.

A drawback of Lockr is that it focuses on social relationships only. It does not define a generic model of authorisation that can be plugged in to various Web applications and provide arbitrarily complex access control for heterogeneous and distributed resources.

### 3.2.4 Invitation-based authorisation

At the recent Terena 2010 meeting in Vilnius [44], Jaime Péres Crespo of the Spanish RedIRIS federation proposed an invitation-based authorisation model [45]. The model is based on the idea of making authorisation user centric and was motivated by the need to make authorisation more scalable.

Today’s federations usually implement attribute-based authorisation. In expanding federations such attribute-based model does not scale because:

- it requires identity providers to continuously add new attributes for their users and to ensure that these attributes are released properly,
- identity providers hold the responsibility for the attributes, making authorisation heavily dependent on them.

In the invitation-based model, the user is invited by the SP or resource owner to get authorised via an e-mail. The e-mail includes a link that triggers an authentication session with the user’s IdP and a subsequent registration of authorisation at a central authorisation engine. The invitation protocol use case is shown in Figure 14.

![Figure 14: Invitation protocol use case (from [45]). SIR is the RedIRIS federation hub.](image)

The protocol has the advantage of allowing authorisation of almost any individual logging in through the federation. As no specific attributes are required to identify the users, the IdPs and SPs don’t have to bother about them.
The protocol works best in a hub-and-spoke based federation (like RedIRIS and SURFfederatie) and is compatible with existing group management approaches.

3.3 Grid-based

Attribute-based access control systems are an active area of research in Grid computing, and several systems have appeared. This section highlights several of them.

3.3.1 OGF recommendations

The Open Grid Forum (OGF) is a community of users, developers, and vendors leading the global standardisation effort for Grid computing. OGF developed the Open Grid Services Architecture (OGSA) [46] to describe an architecture for a service-oriented Grid environment for business and scientific use.

OGF activities are carried out through community initiated working groups, like the OGSA authorisation working group. Its goal is to define the specifications needed to allow for interoperability and pluggability of authorisation components from multiple authorisation domains in the OGSA framework. The group leverages security work that is ongoing in the web services community (e.g. SAML, XACML, and the WS-Security set of specifications) and defines profiles on how these should be used by Grid services. The group identified three functional components to be used in Grid authorisation and is currently working on defining profiles for them:

- **Attribute Authority (AA)** is a Grid service providing reliable information about holder of credentials. An AA assures a coarse grained granularity to identify users in a Grid. The protocol being defined uses SAML.
- **Authorisation Service (AS)** releases authorisation decisions based on users’ and resources’ attributes. The protocol being defined uses XACML and the SAML profile for XACML.
- **Credential Validation Service (CVS)** is an application independent policy engine that validates users’ credentials (digitally signed attributes assertions) to be used by an authorisation service. The protocol being defined uses SAML and WS-Trust.

The interactions between the functional components of OGSA Authorisation framework are shown in Figure 15.
3.3.2 Globus Toolkit

The GT4 AuthZ framework [48] is the last attempt by the Globus team to develop an authorisation service. It is an implementation of the XACML authorisation model, although its master PDP is only responsible for the coordination of external authorisation sources like PERMIS (see section 3.4.2). External attribute authorities like VOMS (see section 4.1.2) can provide attributes.

The authorisation framework of GT4 is based on the ABAC policy model. Many policies use the attributes of the requestor, the service, the resource, the action, and the environment.
GT4 also allows for aggregation of multiple policies from distributed PDPs. In practice, this model is often implemented as a relatively static chain of preconfigured PDPs (and PIPs) that are used to come to an overall decision statement.

### 3.3.3 Akenti

Akenti is an established authorisation service designed to make access decisions for distributed resources controlled by multiple stakeholders. Akenti assumes that all the parties involved in authorisation have X.509 certificates that can be used for identification and authentication. An authorisation policy for a resource is represented as a set of (possibly) distributed certificates digitally signed by unrelated stakeholders from different domains. These policy certificates are independently created by authorised stakeholders. When an authorisation decision needs to be made, the Akenti policy engine gathers all the relevant certificates for the user and the resource, verifies them, and determines the user’s rights with respect to the resource.

The Akenti model consists of resources that are being accessed via a resource gateway (the AEF) by clients. These clients connect to the resource gateway using the TLS handshake protocol, or something equivalent, to present authenticated X.509 certificates. The stakeholders for the resources express access constraints on the resources as a set of signed certificates, a few of which are self-signed and must be stored on a known secure host (probably the resource gateway machine), but most of which can be stored remotely. These certificates express the attributes a user must have in order to get specific rights to a resource, identify the stakeholders who are trusted to create use-condition statements, and determines the attribute authorities who can attest to a user’s attributes. At the time of the resource access, the resource gatekeeper (AEF) asks a trusted Akenti server (ADF) what access the user has to the resource. The Akenti server finds all the relevant certificates, verifies that each one is signed by an acceptable issuer, evaluates them, and returns the allowed access. The Akenti framework is shown in Figure 17.

![Figure 17: Akenti authorisation model in pull mode (from [49]).](image-url)

**Figure 17: Akenti authorisation model in pull mode (from [49]).**
In Akenti several models for authorisation systems have been proposed. One is the pull model, in which the user presents only his authenticated identity to the gatekeeper, who finds (pulls) the policy information for the resource and evaluates the user’s access. Another model is the push model, in which the user presents one or more tokens or assertions that grant the holder specific rights to the resource. In this model, the gatekeeper must verify that the user has the rights to use the tokens and then must interpret the rights that have been presented.

The Akenti policy is expressed in XML and stored in three types of signed certificates: policy certificates, use-condition certificates and Akenti attribute certificates. Policy certificates specify the sources of authority for the resource. Use-condition certificates contain the constraints that control access to a resource. Attribute certificates assign attributes to users that are needed to satisfy the use constraints.

3.3.4 SecPAL

Microsoft’s Security Policy Assertion Language (SecPAL) is a language for expressing decentralised authorisation policies.

SecPAL provides a flexible and robust declarative authorisation language developed for large-scale Grid computing environments. The specification describes the XML syntax and data encoding conventions required to support an implementation of SecPAL [50].

The syntax of some policy languages, such as XACML, is defined only via an XML schema; policies expressed directly in XML are verbose and hard to read and write. On the other hand, policy authors are usually unfamiliar with formal logic, and would find it hard to learn the syntax of most logic-based policy languages. SecPAL has a concrete syntax consisting of simple statements close to natural language. Policies and credentials are expressed using predicates defined by logical clauses, in the style of constraint logic programming. Access requests are mapped to logical authorisation queries, consisting of predicates and constraints combined by conjunctions, disjunctions, and negations. Access is granted if the query succeeds against the current database of clauses. Predicates ascribe rights to particular principals, with flexible support for delegation and revocation. At the discretion of the delegator, delegated rights can be further delegated, either to a fixed depth, or arbitrarily deeply [24].

3.4 PMI-based

Traditional privilege management involves the process of managing user authorisations based on X.509 attribute certificates (ACs). In a way Privilege Management Infrastructures (PMIs) are to authorisation what Public Key Infrastructures (PKIs) are to authentication [51].
PMIs use attribute certificates (ACs) to hold user privileges, in the form of attributes, instead of public key certificates (PKCs) to hold public keys. PMIs have Sources of Authority (SoAs) and Attribute Authorities (AAs) that issue ACs to users, instead of Certification Authorities (CAs) that issue PKCs to users. Usually PMIs rely on an underlying PKI, since ACs have to be digitally signed by the issuing AA, and the PKI is used to validate the AA's signature. Nowadays, there are also PMIs that are based on SAML/XACML technology.

The function of a privilege management service (or authority management service) is to describe authorisation for an individual, making it possible for other systems to make decisions about access. Currently, most authorisation systems use attributes which describe the role of an individual in permanent terms; descriptions such as “staff” are likely to be static until the role itself is changed (e.g. by resignation from employment). However, more and more services and systems emerge which require more complex information in order to determine whether a user is permitted access – or to determine which user out of those permitted access is able to perform some other action, such as editing content. Such information is known as a “privilege”, and an action is known as “function”. Privilege managers also make it possible to place limitations on a particular user when exercising a function, common examples including temporal ("not before 21 June 2005", "only on Tuesdays") and financial ("authorise spending up to 10,000 Euro") ones.

A privilege management system permits central storage and maintenance of this kind of complex authorisation information in an organisation and its distribution to resource providers as required. There are basically three parts to a privilege management system: a service to assign privileges to individuals (e.g. Signet), a service to pass privileges between the assigning service and the resources which will make use of them for authorisation (e.g. Shibboleth, SAML), and a service which makes it possible to check the privileges against the access management policy of the resource (e.g. the PDP component of PERMIS described below).

This section describes several PMI-based solutions.

### 3.4.1 Signet

The Signet Privilege Management System activity addressed the challenges of managing what people can access and giving control of that process to those who make the decisions and was developed by a working group of the Internet2 Middleware Initiative [52].
Signet stores privilege information in its own database, and makes the data available through an API (Application Programming Interface – a documented method for one program to access the features of another). This means that software which uses Signet for authorisation needs to be modified to use the API and to understand the privileges obtained through it, or access privileges obtained from Signet but stored in a different manner. This second method is expected to be how Shibboleth protected services will access Signet privileges, which will be stored in a Shibboleth attribute repository, probably in the form of eduPersonEntitlement values (which will need to be agreed between the service provider and the institution). Protection of the attribute information – the assurance that they have not been tampered with – is given by the security of the database itself.

Signet 1.0 was released in February 2006. The design of Signet is intended to permit close integration with other Internet2 middleware software, particularly Grouper and Shibboleth, while its interface does not rule out use with other similar products.

The Signet project is no longer actively supported by the Internet2 Middleware Initiative. MACE, a group of campus IT architects that helps steer the Internet2 Middleware Initiative, decided to end the Signet project because of lack of adoption. Signet had failed to ensure that an incremental path was available to campuses to help them progress towards implementation of Signet's advanced privilege management capabilities.

MACE's investigation into and development of advanced privilege management capabilities for campus use, however, has continued. The MACE-paccman working group was formed to provide a venue in which major open source projects and campus IT architects could discuss issues and identify common needs related to privilege and access management [53]. Also the Grouper project (see 4.1.4) added role and privilege management capabilities to its roadmap as a means of providing an incremental path to campus adoption of those capabilities.

### 3.4.2 PERMIS

PERMIS [54] is an authorisation server that implements attribute-based authorisation policies. PERMIS is different from Signet in that it is a complete privilege management infrastructure rather than just a privilege assignment manager. It has tools for assigning privileges to users, as Signet does, although they are not as sophisticated. It also has mechanisms for transporting privileges, using LDAP protocol (which is similar to the way in which Signet expects to work with Shibboleth to achieve this).

Additionally, it has an access control policy engine (or PDP) that determines if a user's assigned privileges are sufficient to access a particular resource in the way in which they are requesting. Despite the differences between Signet and PERMIS, the two applications can interoperate, using Signet to allocate the privileges to users, Shibboleth to transport the privilege information, and then eventually using the PERMIS PDP to analyse them and grant or deny the user access to a remote resource.
When a user makes an access request, the PEP authenticates him/her and asks the PDP for an authorisation response, which makes a granted or denied decision. A credential validation service (CVS) validates the users’ attributes that are used by a local PDP. PERMIS requires the Sources of Authority (SoA) to set the policies for every resource they own. The interface to the PERMIS PDP has recently been enhanced to support XACML. This will allow sites to plug in a PERMIS PDP where an external XACML PDP is required. PERMIS is adding support to pull SAML attribute assertions from an attribute authority/Identity Provider or to be pushed SAML assertions by the PEP. PERMIS recognises only two different administrative domains: the VO and the target domain. If this is sufficient for future e-Science applications remains to be seen.

PERMIS can work with Shibboleth attributes or X.509 attribute certificates (X.509 is a standard for digital certificates, here used to prove the reliability of the attribute data). The attribute certificate technology ensures the privilege information is not editable without access to the key held solely by the issuer. While public key certificates are widely used for signing attribute certificates, attribute certificates themselves are not widely used. Thus they require that software using PERMIS attribute certificates is customised to do this, in just the same way that software using Signet will need to be customised to access the authority information through the API. Note however, that use of attribute certificates is not mandatory when using PERMIS, it is just another layer of security that may or may not be required.

The PERMIS language is limited in expressions and semantics compared to XACML. The PERMIS framework does not provide direct support for bilateral exchange of policies and credentials to address privacy issues. PERMIS has in its architecture a subsystem that signs, verifies and validates X.509 attribute certificates used to represent authorisation credentials.

3.5 Summary

There are different approaches to tackle authorisation in a distributed setting. In Web-based settings, authorisation is often based on attributes that are embedded in SAML authentication assertions. This seems to work for most applications. If a more fine-grained access control is required, the XACML policy standard seems the most suitable model for authorisation in identity federations.

XACML, however, is only a starting point that defines the syntax for expressing an authorisation query and an authorisation response. An appropriate back-channel (also called “call-out”) protocol to convey the XACML authorisation queries and responses between identity and service providers could be achieved via the XACML profile for SAML. What is missing is a specific taxonomy for authorisation queries that SPs ask of IdP. This taxonomy is dependent on the business rules that will be established for a specific service. For example, if the user is trying to watch an adult rated movie then the service provider needs to check the user’s age with the IdP before making the content available. The query could be one of ‘What is the user’s birth date?’, ‘Is the user over 18?’ or ‘Can this user watch Mature content?’.
The Grid community mainly uses X.509 certificates and attribute certificates to enforce access. The use of such certificates comes with limitations regarding usage across administrative domains and lack of flexibility. Therefore, the Grid solutions seem not suitable for use in modern identity federations. Moreover, even though the Grid authorisation solutions are kinds of ABAC systems, each solution supports its own policies and cannot support multiple different policies. A more flexible and scalable attribute-based access control method is still needed to achieve more effective access control for the heterogeneous Grid computing environment itself and for other environments.

Privilege management systems haven’t been very successful till today. Signet has failed and PERMIS is marginally used in Grid contexts. Potential causes are the lack of a common standard for the structure of privileges, for storing them, or for passing them between applications. This means that each privilege manager has to invent its own mechanisms for this functionality, and any application wishing to use a privilege manager to maintain the information used for authorisation will need to cater for these mechanisms as part of its software design. This will not help in a successful roll-out of a privilege management system. Moreover, it proofs difficult to determine the user’s entitlements in advance. Something to take into account when considering externalising or federating authorisation.

Interesting to observe is the rapid development of user-controlled authorisation solutions. Driven by Web2.0 developments, UMA, OAuth and similar user-centric solutions pop up like daisies and facilitate access control to user attributes and user-owned content. It is questionable, however, if the user is able to manage his/her policy settings in an intuitive and friendly manner in these solutions. It might become too complex and intrusive for the user. User-friendly policy management tools are required but not available at the moment. Nevertheless, with the increasing demand for more user control in identity federations, further development of these user-controlled access solutions should be seriously monitored.
4 Group-based authorisation

The sharing and coordinated use of resources within large or small, dynamic, and multi-institutional communities is fundamental to an increasing range of computer applications, ranging from scientific collaborations to healthcare. This sharing may involve not only file exchange but also direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource-brokering strategies emerging not only in science but in industry and engineering as well.

Moreover, such sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals and/or institutions defined by such sharing rules form a group or a so-called virtual community or virtual organisation (VO) [55].

Authorisation for access to restricted resources is, in general, granted to groups of users. Being able to manage authorisation on the group level is much easier because there are typically fewer groups than users and because permissions assigned to groups change less often than permissions associated with users. Often, the individuals that are added to a user’s people service list will share some common characteristic such that it will be simpler if they are treated and managed collectively as a group rather than separately. For instance, a user may set access control permissions for her online calendar as 'Let all members of my research team view my experimental data' rather than defining separate rules for each member of the team.

Group formation and access rights management become crucial issues when shared workspaces are used to support collaboration between researchers and sharing of resources. Researchers themselves should be able to form groups and adapt access rights for changing groups and workspaces.

But if a group wants to use several distributed resources, the group manager will have to create duplicated versions of his group multiple times, and send one invitation for each of these resources to all members of your group. Not to mention the fact that managing user permissions in such a distributed manner can quickly become a nightmare. A federated group management infrastructure with support for authorisation or entitlement is something that could help. In such infrastructures, group members are able to federate their services and their identities. This is illustrated in Figure 18.
A key problem associated with the formation and operation of groups is that of how to specify and enforce group policies. Policy enforcement for groups comprising users of multiple institutions and resource providers imposes unique challenges:

- **Scalability.** The cost of administering a group (e.g., adding or removing participants, changing community policy) should not increase with the number of resource providers participating in the group. Resource administration overheads should also be bounded. As each group represents a new policy, it is reasonable to require that the cost of administering a resource should be proportional to the number of groups, and not their size or dynamics.

- **Complexity.** Group policies can be idiosyncratic, i.e., they may apply to sets of resources (e.g., restricting what fraction of total storage capability available to the group), may cross institutional boundaries, and will vary over time. Enforcing these agreements and policies in a distributed fashion introduces difficult bookkeeping issues.

- **Dynamicity.** New members may join the group; others may leave the group. Resource may be added to the group on an ad-hoc basis. The authorisation infrastructure has to able to deal with such dynamic provisioning and deprovisioning of access rights.

- **Provisioning.** Group-related credentials must be provisioned as well as policies that need to base authorisation decisions on these credentials. Doing this across multiple institutions is not straightforward [56].

- **Coordination.** Many different sources of information must be coordinated and brought together for policy decision and enforcement. Users may be members of multiple groups. Combing the right group-related credentials with the right policy requires coordinative effort.

- **Policy hierarchy.** Groups may be hierarchical (i.e. have subgroups). For example, each institution within a collaborating group may wish to define its own institutional policy. Each of these nested policies must be consistent: institutional policy must be consistent with the group policy, which must in turn be consistent with the local policy defined by each resource. Note that group members may be resource owners and define the authorisation policy for their resources themselves.

- **Heterogeneity.** Supporting several styles of access to group membership information. E.g. by invitation, upon request or based on certain entitlements.

- **Privacy.** Group membership may be privacy sensitive.
The solution to these and related challenges lies in the provisioning of mechanisms for distributing policy administration and management and efficient communication of group credentials. The next section presents an overview of several group authorisation solutions.

### 4.1 Authorisation in group management solutions

#### 4.1.1 CAS

A virtual organisation (VO) is a dynamic collection of resources and users unified by a common goal and potentially spanning multiple administrative domains. VOs introduce challenging management and policy issues, resulting from often complex relationships between local site policies and the goals of the VO with respect to access control, resource allocation, and so forth. In particular, authorisation solutions are needed that can empower VOs to set policies concerning how resources assigned to the “community” are used—without, however, compromising site policy requirements of the individual resources owners.

The Community Authorisation Server (CAS) that is developed by Globus provides a solution. CAS centralises authorisation policies that must be enforced by a number of shared resources, minimising the authorisation intelligence needed by the resources [57]. CAS allows for a separation of concerns between site policies and VO policies. VO policies are maintained using the CAS server and CAS administrative protocol.

The central idea behind CAS is that while resource providers can specify a coarse-grained policy, the fine-grained policy decisions can be delegated to the administrator of the VO that is served by CAS. Resource providers grant privileges to the VO and establish a trust relationship with the representative of that VO. That representative then uses CAS to manage the distribution of privileges within the VO.

CAS functions as a "push-model" authorisation service, as shown in Figure 19.

![CAS architecture](image)

Figure 19: CAS architecture. Steps are explained in the text below.
The steps in the figure are:
1. The client, shown at left, sends a signed SAML AuthorisationDecisionQuery request to the CAS server, at right, indicating which resources they wish to access and which actions they desire to invoke.
2. The CAS server establishes the user's identity. Using the identity it determines the rights as established by the VO's policy. It then returns a signed SAML assertion containing an AuthorisationDecisionStatement. This assertion has the user's identity as the Subject and some subset of the user's requested actions.
3. The user presents the SAML assertion to a resource along with an authenticated invocation request. The resource uses the SAML assertion, subject to local policy regarding how much authority was delegated to the CAS service, to authorise the request. The user may use the assertion to potentially make multiple requests, potentially to multiple resources.

Note that it is common for a client to ask for an assertion containing a complete set of rights he may have on a given resource, set of resources, or even on all resources for which a CAS server has authority. Since the SAML statement returned is typically valid for a number of hours, an assertion with multiple rights allows the user to undertake a number of different actions, which may not be known a priori, without having to re-contact the CAS server.

CAS is similar to Shibboleth in its use of SAML assertions. However CAS operates at the level of capabilities rather than attributes; that is, instead of expressing abstractly what someone is, CAS expresses explicitly what actions they are allowed to take. Moreover, CAS operates in the context of VOs, whereas Shibboleth is primarily designed for federations which have a more static character.

A problem with the CAS approach is that the server must know beforehand the details of all local implementations. Other problems with CAS are that it is non-trivial to set up and use CAS [58], and that its centralised model of an authorisation server is likely to have scalability issues when dynamic VOs are to be established or are very large.

CAS can be considered as the precursor of VOMS.

4.1.2 VOMS
The Virtual Organisation Membership Service (VOMS) [59] was developed by the European Data Grid project to allow for attribute-based authorisation to the Globus Toolkit job management services.

VOMS can be considered as an Attribute Authority (AA) focused on VO management. It allows the VO manager to assign attributes (e.g. group or project membership, role possession, or generic key value pair attributes) to users according to their position in a VO. On request VOMS releases signed assertions containing the above described attributes. These attributes are used at the resource level to drive authorisation decisions. Thus VOMS supports dynamic, fine-grained access control needed to enable resource sharing across VOs.
VOMS uses X.509 attribute certificates in a push mode to assert attributes in a modified version of the Globus Toolkit. In the most adopted usage pattern, the attribute certificate is inserted in an extension of proxy-certificates of the users. Due to this construction, however, VOMS is not easily interoperable with emerging Web Services-based technologies in e-Science. VOMS also does not support a pseudonymous mode, nor does it have any other provisions for privacy support. A major weakness of VOMS is its centralistic approach yielding a single point of failure.

Recently, VOMS has been re-engineered to support authorisation standards emerging from OGF and is extended with a standard WS-interface that uses SAML. The certificate and SAML based VOMS architectures are shown in Figure 20.

Figure 20: VOMS server releases Attribute Certificates (a) or signed SAML assertions (b).
From [60].

4.1.3 G-PBox

G-PBox is a gLite component and aims to solve the tasks of policy management in a production Grid environment.

G-Pbox is the policy handling service designed to manage resource allocation and sharing. The service interacts with VOMS, services and resources to execute jobs and allocate storage according with the site, VO and Grid policy. Resource managers at each site interact with the service to grant/withdraw resource access according to the site policy inside the wider resource management of the whole Grid.

The G-PBox architecture is based upon a composition of modular objects, Policy Boxes (PBox), as shown in Figure 21. There are PBoxes at various levels: VO level, Domain Level, Site level, Farm Level, possible sub-Farm levels, etc. This helps to limit the scope of a particular group of policies. The VO PBox is the authoritative source for VO-wide policies, the Grid PBox is the authoritative source for Grid-wide policies, while the Farm PBox is the authoritative source for policies specific to a particular farm.

---

6 gLite is the next generation middleware for grid computing, see [http://glite.web.cern.ch/glite/](http://glite.web.cern.ch/glite/).
Each and every client that wants to be policy-aware, i.e. become a PEP, has a configured PBox that will be contacted whenever a policy decision is required. The PBox will at this point make its own decision and communicate it back to the resource.

A PBox itself consists of several modules (see also Figure 22):

- **PDP** - The Policy Decision Point (PDP) is the module that receives requests for policies and sends back responses. It uses the XACML 1.0 language to express both policies and requests/responses.
- **PAT** - The Policy Administration Tool (PAT) is the module that the administrator of a PBox normally uses. It is used to create, review and send policies, and to take decisions on policies received from other PBoxes.
- **PR** - The Policy Repository (PR) is the module that stores all the policies, both locally-created or received from remote, along with such information as status (accepted, refused), origin (from which PBox it originated), etc.
- **PCI** - The Policy Communication Interface (PCI) is a layer that surrounds the PBox and is used for all its communications with the outside, whether they are other PBoxes or PEPs. The communication itself may be unencrypted or protected by the Globus Security Infrastructure (GSI), depending on the configuration. By default, communications between PBoxes are secure, confidential and mutually authenticated, while communication between a PBox and a PEP is unencrypted.

A G-PBox must have knowledge of all the applicable policies, whether they are local or external. For this reason, a PBox can send new polices to the PBoxes at other layers, and those PBoxes may immediately accept them or wait for the administrator confirmation.
Policies are defined in XACML and the framework uses the XACML request context in the communication between PEP and PDP in order to obtain authorisation decisions. The XACML request context is transported to the PDP in a SAMLv2 request message. The XACML response context contains an authorisation decision and optional obligations that must be enforced by the PEP, either before, with or after enforcement of the user’s request.

### 4.1.4 Grouper

Internet2’s Grouper Groups Management Toolkit enables project managers, departments, institutions, and end-users to create and manage institutional and personal groups [61]. Grouper keeps the group membership decisions in the hands of the group owners and access control in the hands of the group resource owners.

Figure 23 shows how Grouper integrates with the local ICT infrastructure. Important elements are the LDAP provisioning connector for the synchronisation of group memberships with LDAP user objects, the local Grouper database that contains all group data and the Grouper Client to enforce group membership access to the application.

Grouper is not federation friendly, and does not facilitate having members from outside the own institution. The centralised group management approach of Grouper works well inside an organisation but suffers from lack of support for multiple registries in multiple organisations. Federated group management may solve this, as is described in [62].

Ideally, group membership should be abstracted from the services so that membership information is “globally” available to any service. For instance, the same group of students cooperating as a Sharepoint team should also be able to use a media service for sharing content among the group members without having to recreate the group in the media service (which is today’s practice). The group-attribute approach that is described in the next section facilitates this ambition.
4.2 Group attribute-based access

Authorisation for large groups like a VO is manageable for users that share common attribute values. For instance, it is relatively simple to allow all medicine students access to a particular service. In a more common authorisation scenario, however, VO members never share a common attribute.

A possible approach to tackle this challenging scenario is to give members of a VO a common attribute. This group or VO attribute represents membership. The problem of setting these group attributes is that in a federation the home organisations (i.e. IdPs) are responsible for the data they assert and that they don’t set group attributes on request. So, setting group attributes at the IdPs is infeasible. A solution to this problem is to set the attributes elsewhere, for instance at the hub of the federation or a third party like a VO platform, and make SPs fetch them from these locations. SURFteams, for instance, operates in this manner and serves as a group attribute provider for SURFmedia. But also the Swiss federation SWITCH has already prototyped this approach (more information can be found here [64]). The SWITCH approach is shown in Figure 24.

An important aspect to take into account is that consistent user identification is involved at all parties involved (IdP, SP, and group attribute provider). This shared identifier is used in the SAML2.0 Name Identifier of the attribute request.
The enrolment of group attributes, i.e. group memberships, can be based on self-registration or invitations (via e.g. e-mail like is used in SURFteams).

The advantages of this approach is that no additional protocols are required (it is pure SAML2.0), it is simple to configure and scalable. Moreover, it is suitable for inter-federation setups as well. The disadvantage is that SPs need to define specific group rules in their policies. Note that these SPs may very well be the group members themselves; they make resources available to the group. Another problem that may arise is when users participate in multiple groups; somehow the right group attribute needs to be selected before handing it over to the PEP of the resource.

4.3 Summary

Identity federations have emerged to make user attributes available across organisational boundaries (and even across federations). The next logical step will now be to make these attributes available to resources for both user management processes within groups and for authorisation purposes on resources. Consequently, this would lead to two distinct attribute authorities participating in the management of user attributes: the traditional group management systems such as VOMS or SURFteams and the user’s home organisation’s IdP. These two authorities issue different kinds of attributes:

- Home organisation related attributes, i.e. user attributes managed by their respective home organisation’s IdP. They identify and describe the user by e.g. stating his name, nationality, telephone number, his affiliation to organisational units, and his roles within these units, e.g. professor at a faculty or student of a certain study course. These attributes are managed and issued by the IdP.
• Group attributes that describe users by their memberships, roles, and capabilities he has within a group. These attributes are managed and issued by a dedicated group management system such as VOMS.

Somehow these attributes need to be aggregated properly and provided to the SP. The SP then can enforce access to group-related information. Prerequisite is that the SP is able to deal with these attributes in his policy. This is not likely to happen in distributed settings. An alternative approach is to make use of multiple policies and combine them such as is done in CAS.

Also notice that another classification of attribute types can be made that somehow resembles the above-described types: attributes used for authentication purposes and those used for authorisation purposes. The former type includes attributes that are closely related to the user’s identity and are often provided by the IdP; the latter type includes attributes that are often application specific and may very well be provided by authorised third parties, i.e. attribute service providers.
5 Authorisation challenges in federations

Building secure and dependable authorisation systems for federated and multi-domain computing environments poses numerous challenges. Below are a set of issues and requirements that are based on the use cases and shortcomings of the technological state-of-the-art and need to be addressed in such environments.

5.1 Generic challenges

5.1.1 Heterogeneity of systems

The authorisation infrastructure should be able to deal with a highly-distributed environment of possibly heterogeneous subjects and objects. It needs to address the heterogeneity of components that comprise such an infrastructure and it should provide means to ensure interoperability between them. Components should be able to exchange information meaningfully when making authorisation decisions.

Authorisation decisions that span administrative domains require that components in every domain are capable of correctly producing, accepting and interpreting authorisation information from a group of potentially heterogeneous peers. A common agreement protocol, syntax and semantics of every piece of information that is exchanged between components of the authorisation system is a necessity. This includes interoperability at the level of language that is used for permission specification and at the level of protocols that are used for communication between various components of the system.

However, there are many applications running in a federation, and many of these applications manage their own policies, and do it differently. This makes access control very difficult to manage. Integration of such policies entails various challenges including reconciliation of semantic differences between local policies, secure interoperability, containment of risk propagation and policy management. How to deal with authorisation in heterogeneous federations with applications that may not even be compliant to federation protocols such as SAML, WS-Federation or XACML?

Possible solution directions are:

- The deployment of gateways that are able to deal with authorisation systems heterogeneity. Axiomatics for example provides multiple XACML-based PEPs for a number of XML gateways and for application servers such as J2EE and .NET ASP. The latter PEP also has a support for Microsoft Active Directory Federation Services 2.0 (earlier called Geneva) claims. The Geneva modules gather claims and translate them into XACML attributes which are then, together with access requests, submitted to a PDP for access decisions [65]. Other examples are Exostar’s Enterprise Access Gateway [66] and the proposed gateways in eduGAIN for providing inter-federation attribute translation.
• To tackle differences in semantics between policies by providing a uniform representation of access control rules from different domains. This can be done using meta-policies which would mediate each access request that span multiple domains. Another approach would be to enforce usage of a standard policy language such as XACML that would be used consistently throughout the entire security system. As organisations are moving towards standardising their authorisation policies, the latter approach is more favourable.
• To externalise authorisation from the actual applications thereby removing local application heterogeneity. Instead of implementing access control policy, applications should use an external and potentially centralised access control system in order to make the decision regarding access control policy. XACML would be a suitable candidate for policy externalisation.
• To create PEP interoperability via a middleware layer. The OpenAz project is working on such a middleware layer (see below).

5.1.1.1 OpenAz

There is an increasing consensus that access control decisions should be externalized from applications or services to a policy engine implementing a PDP. To take full advantage of this model, one needs to embed PEPs in applications, middleware and services in a performing and flexible way. For instance, the use of XACML requires changes to services to make the appropriate XACML authorisation queries. In other words, the service needs to become a PEP. This is addressed in the OpenAz open source project of Open Liberty [67]. OpenAz aims at creating language bindings for the XACML PEP request-response protocol. Figure 25 shows the OpenAz architecture.

![OpenAz Architecture Diagram]

Figure 25: OpenAz architecture.

The motivation for the OpenAz project is to facilitate the development of a standard interface framework, which will provide a well-understood common basis for:
1. externalising authorisation from applications
2. supporting use of authorisation policy engines
3. building PEPs, based on XACML foundations, that can be created in a variety of language and technology contexts
4. integrate existing authorisation infrastructure by abstracting it within a XACML context in order to develop an authorisation uniform framework
5.1.2 Discovery of PDPs, policies and attributes

Once an access request is made to a resource, the PEP needs to contact the PDP to determine whether access should be granted or not. The PEP needs to know which PDP it must use. In systems with a limited number of components of the authorisation infrastructure the relationship between PEPs and PDPs can be static. When the PEP is initialised it simply checks whether a predefined PDP is available to instantiate a communication channel with this point. Such static relationships between components of the authorisation infrastructure do not scale well but are easy to design and implement.

Although static binding between enforcement and decision components in small distributed systems is sufficient, it does not fit into large computing environments spanning multiple separate administrative domains. At first, PEPs may delegate rights to other domains and may not wish to specify exactly which PDPs should be used. Such PEPs may just be satisfied with any decision that is signed by a particular administrative body. Moreover, in case of large and dynamically changing distributed systems, a static binding between PEPs and PDPs may not be feasible. In such cases a discovery mechanism needs to be employed.

PDPs on the other hand must be able to find the right policies to be used and the corresponding set of attributes that is needed according to the policies. Finding policies can be based on file names or pointers to policy information points. Policy selection or resolution is typically done based on authorisation request parameters such as resource or subject attributes. The home IdP of the user may not have all the attributes. For instance, it will not have group-related attributes. Multiple attribute service providers may need to be contacted by the PDP in order to aggregate the required set of attributes. Knowing where to find the right attribute service providers requires proper configuration management by the federation operator to simplify the discovery process. See also section 5.2.3 for more details on attribute sharing.

Another issue is that, unlike books and ISBN numbers, there is no globally unique identification system for e-Science data/content resources. Each institution and experimental setup/program has its own classification and identification system. This becomes problematic if the SPs need to make authorisation decisions based on the specific content each user is attempting to access. What to do in a federated authorisation model where the object or resource (i.e. the program) is known by different identifiers? The definition of a neutral identification scheme for all scientific content would be ideal but does currently not exist.
5.1.3 Communication performance

Authorisation decision is often based on information from different highly distributed components. Such a situation occurs when PEPs reside in different domains than PDPs, that additionally collect information from a distributed set of PIPs. In such scenarios it is necessary to ensure that communication between components of the authorisation system is efficient in terms of the number of messages that are sent between components and the size of those messages.

One of the approaches of minimising the number of interactions between components of the system is the use of caching. Enforcement points may cache decisions made by PDPs. Additionally, PDPs may cache policies that they would normally retrieve from administration points. Caching, however, comes with the risk of outdated policies and subsequent unauthorised access.

Another approach to minimising the number of messages that must be sent between components of the authorisation system is to use policy syndication as proposed in [68]. A federation Policy Administration Point (PAP), which is managed by a central authority, i.e. the federation operator, may hold a federation authorisation policy. Such a policy is then syndicated to more local PAP components residing in different administrative domains or in the same domain in which syndication takes place. A hierarchy of such PAP interactions can be created as depicted in Figure 26.

![Policy syndication hierarchy](image)

Figure 26: Policy syndication hierarchy.

Federating policies is another approach to tackle the distributed nature of authorisation across multiple domains. The concept is shown in Figure 27. Basically, it allows two or more parties to negotiate and combine their policies into a common, federated policy document.
Communication between components of the authorisation infrastructure should also aim to be based on lightweight messages that would not affect the overall throughput of the computing environment. Because XACML uses XML to encode access control policies then the size of policies and privilege statements is significant due to the XML encoding overhead and verbosity of the language.

5.1.4 Session-based authorisation

When initiators perform a series of operations on resources, the authorisation decision-making should be made as efficient as possible, so that quick decisions can be made. Information that is common to each authorisation request, such as the initiator’s details, should only need to be sent once to the authorisation service. The use of session identifiers may facilitate such efficiency. The challenge is to create a common understanding of these session identifiers across multiple domains.

5.1.5 User oriented authorisation

Users may want to control access to their own data. Facilitating this in an easy and understandable way is challenging, as access rights for collaborative systems tend to be rather complex, leading to difficulties in the presentation and manipulation of access policies at the user interface level. To attain this goal of user-friendly policy management, attribute-based policy specialists will have to work with experts in HCI to develop an understanding of human-centric policy management. This may include intuitive user interfaces, explanations and justifications of policy decisions, policy analysis algorithms and engines, and ways to reason about distributed authorisation decisions [69].

5.1.6 Groups and collaborations

A key problem associated with the formation and operation of distributed virtual communities is that of how to specify and enforce community policies. Group or collaboration-based authorisation is particularly challenging due to its dynamicity, i.e. members join and leave the group continuously. The authorisation infrastructure must be able to deal with this dynamicity in a scalable and user-friendly manner.
The use of attributes for group representation is relatively easily achieved in modern SAML-based federations. However, the provisioning of group-attributes in service provider policies is not trivial and problems occur when users participate in multiple groups.

One of the problems with SAML based attribute assertions is that these are primarily focused on persons. An attribute is always in some way a property of a person. This relates poorly to the need to exchange other data between the VO and SP, like e.g. group related data or notifications. Also attribute assertions will most likely not scale very well if the amount of content in the assertion becomes larger. Therefore a mechanism that can be provided with a fairly limited set of person attributes in the SAML assertion (only a person identifier?), and then use that attribute to query additional data from the VO seems better suited. The OpenSocial Social Data API can be just that API. At the moment experience is gained in Géant3 with the exchange of attributes between the VO and SP based on the REST-based OpenSocial API.

5.1.7 Access control to authorisation decisions

For reasons of security and privacy, authorisation services should be capable of enforcing access control on who can request authorisation decisions. In the simplest incarnation, authorisation services should be configurable so that they only answer to queries from a set of trusted target resources. More complex implementations could allow for finer-grained policy based on the initiator and request. Some implementations even require proof that an initiator requested an action prior to authorising it. Managing such access control in multi-domain settings is not trivial. The new delegation functionality in the soon to be standardised XACML3.0 specification may solve this issue as it allows policy managers from multiple institutions to co-specify authorisation policies.

5.2 ABAC specific challenges

In highly open systems like federations, with a large variety of services and a high number of potential users, existing identity and role-based access control models are not flexible enough. In such settings, attribute-based access control (ABAC) has proven its appropriateness. This is reflected in the utilisation of ABAC in most of today’s federations. However, specification and maintenance of ABAC policies has turned out to be complex and error-prone even in federations of limited size, especially if heterogeneous attribute schemes are involved. This section discusses several ABAC-related challenges.
5.2.1 Understanding attributes - formats and semantics

The attributes a user possesses do not necessarily match those specified in the authorisation policy of the resource. In a federation it is therefore desired that authorisation policies somehow support a mechanism to transfer attributes (as well as credentials) of federated users across administrative boundaries for them to obtain access to federated resources according to the applicable policies. Such credential federation demands for interoperable protocols that allow federation members to communicate user attributes (and credentials) in a uniform and unambiguous manner. Standardisation of attributes or attribute mapping in a straightforward manner or via intelligent use of semantic inference may provide solutions for overcoming attribute interoperability.

Standardisation involves the definition of a common set of attributes. This allows the attributes that are issued by every issuing domain to be understood by each receiving domain. This is a feasible approach for federations. Perhaps the most extreme example of this is credit cards such as Visa. Visa cards (a plastic credential) are issued by hundreds of different issuing banks, but all contain the same Visa logo (attribute) and all are treated as being equal by the relying parties, regardless of the issuer. The US academic community also adopted this approach some years ago, by defining the eduPerson schema [70], which is a collection of standard attribute types, and in some cases (for the affiliation attributes) standard values as well. However, in the general case it will never be possible to define standard sets of values for all attributes that will provide sufficiently fine grained access control both for and between all organisations in a federation. Locally defined values of standard attributes will become the differentiating factor for fine grained control.

While standardisation efforts such as eduPerson [70] and SCHAC [71] provide a framework for compatibility, different needs, understandings and societal arrangements are likely to get in the way of perfect interoperability.

For instance the eduPerson staff-attribute does not mean the same at all European universities. Sometimes it means that a person is a member of the faculty (e.g. the professor but not the gatekeeper) and sometimes it means that he is not a member of the faculty (e.g. the gatekeeper but not the professor). Things get even more complicated when a user is a student, staff member and employee....

The alternative approach to standardisation is to define attribute mappings. The function of attribute mappings is to map external (unknown) attributes into internally known ones. With this approach the unique combination of issuer, attribute type and attribute value are mapped into a locally understood attribute value or role, and the remote users with these credentials inherit the permissions that are granted to the local attribute value or role. Of course, this requires the mapping function to be configured such that it can map new types of attributes, but this is a tractable problem. The use of ontologies or the intelligence of the semantic web may provide a solution for this problem [72].
Most attributes, however, have a common understanding and many services only need the member-attribute which enjoys such a common understanding. For the other attributes, it seems advisable to determine their importance and test the definitions of the most important ones. Problems should be documented for others’ benefit.

Another approach to address the lack of attribute harmonisation is to use semantics. Putting user, resource and environment attributes into a semantic context helps simplifying the specification and maintenance of policies [73]. Essential elements in a semantic web based approach are a standardised metadata model for describing resources, a standardised syntax, and a standardised vocabulary (a so-called ontology).

Using XML, resource metadata can be encoded using the Resource Description Framework (RDF, [74]) as the language for the description of resources and RDF Schema (RDFS, [75]) for the definition of the metadata schema. The Open Document Format (ODF) for instance is based on RDF and allows for representing document resources such as spreadsheets, charts, presentations and word processing documents. Other standards for the description of e.g. scientific data or sensors are not available.

An ontology is capable of describing concepts, e.g., persons that exist in a certain domain and relationships among them. These concepts can then also be used as a vocabulary for the metadata of resources. The standard for describing ontologies is the Web Ontology Language (OWL) [76], supporting richer semantics on top of RDFS.

The next step involves processing and analysis of ontologies, i.e. drawing conclusions and gaining new information through combination. Implicit information in the data can be made explicit by using so-called reasoners or inference engines. Simple inferences are already possible with RDFS and OWL, for instance through inheritance. More complex custom inference rules require the usage of a special rule language. A promising approach based on the Rule Markup Language (RuleML) is the Semantic Web Rule Language (SWRL, [77]). Integrated tools for managing RDF(S) and OWL data are available (e.g., Jena5 and Sesame6). These tools also provide simple inference capabilities as well as query languages like SPARQL [78]. An approach for ontology-based authorisation based on OWL, SWRL and SPARQL is described in [79].

Recent initiatives show that standardisation and publication of attribute formats and semantics is taking place in several sectors. The Open Geospatial Consortium (OGC) has developed specifications relating to access control of geospatial data. Healthcare standards have been profiled for the implementation of access control in healthcare. Work is occurring in the OASIS XACML TC to specify attributes for US Export Control and for Intellectual Property. Semantic web technology could play a role in better integrating authorisation attributes.
5.2.2 Attribute assurance - Trustworthiness of attributes

There are situations in which a service provider doesn't really need to know the identity of a person, but just needs to know some particular attribute(s) about someone. For instance, is the person over a certain age, or does the person belong to some particular organisation or other category of individuals? In this case the relying party needs assurance about these attribute values. This is called "attribute assurance".

The big question in ABAC is how to judge the "trustworthiness” and/or "authoritativeness” of each attribute that are used to make the decision?

Elements of identity attributes are known to differing degrees depending on how they were acquired by the identity provider, protected from modification, and periodically refreshed. Several levels of attribute assurance can be distinguished (from low to high trustworthiness):

- Attributes obtained from an unverified third party;
- Self-asserted attributes;
- Attributes from trusted credentials such as passport and driving license;
- Attributes from trusted third parties or verified from credentials;
- Attributes resulting from business processes that verify data;
- Attributes that are assigned by the IdP or its organisation.

Other parameters that determine attribute assurance are the ways by which the attributes are stored, managed, accessed, and conveyed between an IdP and SP. Specifications and vocabularies for describing assurance levels of attributes in federated access management environments are not available.

It is possible that two different sources of the same attribute, even if they have the same value, can engender totally different levels of assurance in that value. The 'how you got it 'matters here: A date of birth gathered during some questionnaire or a date of birth certified by an authority carry different business weight. As yet, there is no standardised syntax by which these differences might be described. What is needed could be something equivalent to SAML's Authentication Context but for attribute statements and not authentication statements.

In SAML the LoA can be communicated as an attribute. Even better is to use the SAML Authentication Context ability to convey the LOA statements in conformance with an identity assurance framework [80]. A similar “Attribute Context” approach for LoA regarding attributes is lacking in the SAML specification. An Attribute Context model has been proposed in [81]. The proposed extensions integrate nicely into a standard saml:AttributeStatement and convey the metadata about individual attributes to an SP that can make a more nuanced access control decision.

Another approach is taken by Chadwick et al. in [82]. Built on NIST’s concept of assurance levels, they propose to have separate metrics for identity proofing processes (expressed in the Registration LoA) and the authentication of a subject (expressed in the Authentication LoA). Registration and Authentication LoAs are combined to a Session LoA and sent in each assertion from an IdP to a SP.
In the recent USGov/Internet2 Tao of Attributes workshop attribute assurance was a major point of discussion [83]. Some participants proposed an "asserted-by" tag solution, whereas others suggested an "issued in conformance with profile urn:xxx" tag.

Though the Kantara Initiative is working on Level of Assurance regarding the trustworthiness of identities [84], its focus is primarily on authentication and less on the attributes associated to an identity. For the moment, Kantara has parked attribute assurance as a future workstream.

OGSA’s Credential Validation Service (CVS) is currently mainly used in Grid environments but may very well become an important functional component in future e-Science infrastructures for the validation of attributes. A CVS can very well be outsourced to a domain-local third party.

Another, less technical approach, regarding attribute assurance is to professionalise IdPs in the federation. Setting and governing high-quality levels regarding identity and attribute management by the IdPs may help improving the overall trustworthiness of the attributes that are asserted by the IdPs. Auditing is an excellent tool to enforce and guarantee high-quality levels.

Furthermore, if the SP cares about the reliability of one or more attributes it can deal with that in the contract that it engages in with the IdP/Attribute Provider/Federation Operator. In other words such a SP does not treat all IdPs equal.

5.2.3 Attribute sharing - ArisID

Every application has the same problem: it needs particular information about individuals (identity attributes), and there are many ways to get them. One of the most common protocols to look up identity information is LDAP, and many attributes about users are stored in LDAP directories. But this is not always case. Identity information can just as well be stored in relational databases. In a collaborative or distributed environment, information can come from many other sources, SAML assertions, web services lookups.

The goal of Open Liberty’s Aristotle project is to create an open source programming interface that provides a standardised access to identity service libraries also known as "ArisID providers". It can be understood as a comprehensive framework for applications to declare their identity requirements and have them fulfilled without having to worry about looking up individual identity attributes from specific sources.

ArisID is designed for developers to access identity information using a single API. ArisID enables access and management of identity information stored in different types of repositories accessed using different protocols.
ArisID puts two key aspects of the Identity Governance Framework (IGF) in practice: a declarative way for applications to make their requirements known and have them catered for, plus CARML - the Client Attribute Requirements Markup Language that defines how this declaration is done. ArisID is an API for accessing and managing personal or identity related information using CARML as an XML data model. In addition to being useful from a privacy perspective, CARML enables important new developer features:

1. The ability to automatically generate a data model in the form of Java beans.
2. The ability to use sophisticated data providers that can connect applications to personal information sources using multiple protocols and virtualisation.

The use of the CARML data model does not assume the pre-existence of a particular database or LDAP schema. Instead, a developer is able to create an application specific data model and write code as if the data model were a straight forward database. Then, at runtime, the provider layers of the API can be configured to connect to many different types of data repositories and network configurations including multiple directories or databases. This way the burden of determining the how (i.e. LDAP, SAML, OAuth, etc.) and from where (i.e. dealing with discovery) to obtain the attribute is taken off the application, and assumed by the identity infrastructure.

5.2.4 Conditional replies and obligations

In an attribute-based authorisation infrastructure, authorisation decisions should not only express permit or deny statements, but also be able to facilitate conditional policies in situations where the authorisation server may not have sufficient information to make a decision.

A possible approach to deal with such conditional policies is to make use of so-called obligations (see reference [85] for more information). Authorisation results from the PDP may come along with obligations that the PEP has to fulfil as part of the authorisation request. Obligations can be specified in the policy and assign values to variables that are returned along with the result – a policy may have one set of obligations that is returned if its result is “Deny” and another set that are returned if its result is “Permit”. For example, if the result is “permit”, a policy may return an obligation saying the accessed data must be destroyed within 30 days. In an XACML-based infrastructure, the PEP that invokes the XACML evaluation engine is responsible for carrying out obligations. Basically this implies that the PEP does a best-effort at an obligation. If it is not able to perform an obligation, it does not mean the access is denied. Enforcement of the obligations is therefore difficult to achieve.
5.3 Policy management related challenges

Policy-based authorisation systems are becoming more common as information systems become larger and more complex. In these systems, to authorise a requester to access a particular resource, the authorisation system must verify that the policy authorises the access. The overall authorisation policy may consist of a number of policy groups, where each group consists of policies defined by different entities. In order to make the right authorisation decisions, it is needed to select and find the right policies. Each policy contains a number of authorisation rules. The access request is evaluated against these policies, which may produce conflicting authorisation decisions. To resolve these conflicts and to reach a unique decision for the access request at the rule and policy level, rule and policy combination algorithms are used. In the current systems, these rule and policy combination algorithms are defined on a static basis during policy composition, which is not desirable in dynamic systems with fast changing environments. In order to create more dynamicity, the context should be taken into account as well [86].

5.3.1 Policy integration and interoperability

Many distributed applications such as dynamic coalitions and virtual organisations need to integrate and share resources, and this integration and sharing will require the integration of authorisation policies. In order to define a common policy for resources jointly owned by multiple parties, functionality may be required to integrate policies from different sources into a single policy.

Multiple policies may be needed to evaluate the request of a user, which also requires the support for combining rules from multiple policies to support composite policy evaluation. This entails that the policy is specified using a standardised vocabulary to allow multiple domain policies to seamlessly interoperate. The use of a standardised meta-model is essential for clear elicitation and integration of interoperable policy specification across the federation. One such meta-model is proposed for multi-domain network resource provisioning [87]. The paper proposes and describes an XACML-NRP policy and attributes profile for network resource provisioning.

Interoperability between policies should be promoted in order to provide common semantics and vocabularies for enforcement. The XSPA profile of XACML for instance facilitates this by describing mechanisms for authenticating, administering, and enforcing authorisation policies that control access to protected information residing within or across administrative boundaries in the health domain [30].

XACML for instance supports algorithms for policy combining (see Figure 28).
Since an XACML Policy may contain many Rules, and since various Rules might return different values for the same access request, there must be a way of resolving conflicts in their results. This is done by specifying a "Combining Algorithm" for each Policy. There are several standard Combining Algorithms, but these are extensible. A typical Combining Algorithm is "Deny Overrides": if any Rule returns "Deny", then the Policy returns "Deny". Only if all Rules return either "Permit" or "Not Applicable" does the Policy return "Permit".

Just as multiple rules can be combined in a policy, multiple policies can be combined in a policy set (see Figure 29). Policy sets can also contain other policy sets, allowing arbitrarily deep nesting.

In more complex scenarios with multiple distributed PDPs, matching of policies is challenging and may require standardisation. But matching by itself may not seem to be sufficient; mechanisms to create the resulting agreement have to be drawn up, including a feedback mechanism in case the matching fails to produce an agreement and the parties wonder why.
5.3.2 Policy conflict resolution

Policy inconsistencies may result in illegal access to resources or in legal access being prevented. Therefore, ensuring that policy conflicts are resolved is an important issue.

Though XACML specifies the use of rule and policy combining algorithms - which are used for making authorisation decisions based on policy sets and multiple rules - it is not designed for conflict resolution. For instance, the SunXACML reference implementation returns “not applicable” in case of a policy conflict. Conflict resolution is addressed externally to XACML and is described in a number of works.

5.3.3 Policy modularity and reusability

The collaborative nature of a federation requires the specification of semantic and contextual constraints to ensure adequate protection of federated resources. Semantic constraints include high-level integrity principles that need to be captured in the authorisation policy, such as Separation of Duty (SoD). Contextual constraints include temporal or other environmental attributes surrounding an access request that must be evaluated to decide on resource provisioning (e.g. federation context, trust anchors, session IDs, etc.). Supporting semantic and contextual constraints in the authorisation policy, however, is not trivial. Most of the existing policy management approaches have tightly coupled these constraints with the policy which is not recommendable in a dynamic federative environment. This is because the integrity requirements and contractual obligations within a federation might change on-demand, and an access management system must be flexible enough to facilitate such adaptation. The policy design must therefore be based on well-understood principles of modularity and reusability that allow independent yet interoperable specification of various components of the policy.

5.3.4 Policy provisioning

One of the goals for policies is to allow dynamic updates to policies by policy decision makers without the need for a software/technical person having to do something to incorporate the policies into the federation. It will be more difficult to achieve this goal if the PAP is not separated from the PDP. Having a centralised policy database allows for more efficient policy provisioning strategies.

5.3.5 Policy creation and analysis

The definition of policies in most policy languages is overly complex, cumbersome, and time consuming. This may result in syntactic and semantic errors. XACML is a good example of such a flexible but complex policy language.
The automated generation of XACML policies from a modelling notation that is more accessible by humans than directly editing XACML policies may speed up the whole policy engineering process. It allows for the direct definition of policies in the context of the underlying business process, avoiding potential inconsistencies between defined security policies and process models based on model changes. User-friendly policy generation tools, however, are barely available and often still a topic of intensive research.

Tools for analysis of policy languages like XACML are not simple to implement. There are two reasons for this problem. The first is that the structure of the language often makes it more difficult than traditional authorisation models and simple, less practical policy languages. The second is that many existing tools are designed to operate in limited environments, i.e. in enterprise settings.

The usual request for analysis is to report what users are allowed to do under certain circumstances. However, this kind of report is infeasible in federations for several reasons. Many query mechanisms cannot be reversed. For example, SAML will provide the attributes of a user, but there is no easy way to ask for all the users with a given attribute value. Even where this is possible, not all the future sources of attributes may be known. Finally, a query may take so long that the data may have changed while it is in process.

Possible approaches to tackle these problems are to partially evaluate policies to a form where they are relatively easy to comprehend, to define sets of inputs and run a whole series of decisions over all the combinations, or to reduce policy languages such as XACML to a logical language by factoring out constructs used to extract and format attribute values [88].

5.4 Summary

Effective mechanisms for authorisation in federated systems must take into consideration the access control requirements of protected resources as stated in their local policies of each participating member in the federation. However, the requirements for federated information sharing for authorisation purposes are not only significantly more complex, but also inherently heterogeneous across the multiple federated sites. The key challenges here include specification of authorisation rules, the design of interoperable federation protocols, and specification and enforcement of authorisation constraints.

Addressing these challenges in the access control policies while overcoming the complexity and heterogeneity calls for a sophisticated policy engineering methodology, that can be adopted for composing federated access management policies that can not only adequately capture the access management requirements but also interoperate across the multiple federated sites. Additionally, given the scale and depth of modern-day distributed systems, it is imperative that the methodology is based on standardised constructs that can be readily integrated into existing systems. This requirement becomes increasingly important when the deployment environment is a federated system.
Policies across multiple systems must be designed in such a way that they can be uniformly developed, deployed, and integrated within the federation. This is a challenging ambition, but one that may help tackling most of the aforementioned challenges in federated authorisation.
6 Conclusions

One of the core challenges in e-Science service infrastructures is to enable coordinated resource sharing among dynamic collections of individuals, institutions and resources. From a security point of view, each resource owner is strongly interested in accurately setting and managing the authorisation policies on its resources. This has to be done at multiple levels: the user, the group or virtual organisation, the institution and/or the federation. Achieving such multi-level authorisation effort is not simple.

An important observation is that authorisation remains an area without a dominant standard, and which presents substantial usability problems to resource owners and users.

Fine grained authorisation is increasingly desirable but difficult to implement in an identity federation such as SURFfederatie. The ABAC model seems the most promising solution as it offers flexibility and allows for fine-grained authorisation that is based on attributes. The use of attributes, however, comes with a variety of challenges relating to the attribute information required to make policy decisions. These challenges are related to the lack of standardisation of their formats and semantics, trustworthiness, and distributed nature.

A policy-based authorisation strategy suits the ABAC model best and fits very well in SAML-based federations. Policy management standards that externalise authorisation (e.g. XACML) are also desirable, but are not widely deployed. Complexity in adapting applications to take advantage of such standardised and externalised authorisation methods will continue to delay adoption. Privilege-based authorisation solutions such as Signet or PERMIS try to achieve externalised authorisation by centralising privilege management. These solutions seem not suitable for use in federations, as it will be difficult to define in advance all privileges for the numerous applications that are offered in a federation.

Many SAML-based federations have adopted a push model. This means that authorisation attributes are pushed by the identity provider to the service provider. This may seem efficient but results in possible unnecessary communication of user-related attributes. A pull-based model would be more preferable and would better fit with the federated authorisation model and the use of the XACML standard. In such a model, policies are provisioned push-wise and attributes are pulled from the IdP. Pull-centric identity management is just beginning to emerge (e.g. Axiomatics, Oracle Entitlement Service and MS Active Directory Federation Services v2). But it doesn't have a lot of momentum yet, particularly not in federations.
User-controlled authorisation solutions such as OAuth and UMA are driven by social media and gaining more and more traction. Allowing the user to manage his/her own attribute release policies or resource access policies requires user-friendly and understandable management tools and/or efficient user consent interactions. Such tools or interactions are not yet available. Since most of the federations for research and higher education are IdP-centric the case for user-controlled authorisation may have to wait until more mature user-centric federation models arise. One application that may drive the need for user-controlled authorisation is the sharing of resources in groups.

Combining federated identity management with group management seems promising. Such a combined approach leverages the efficient authentication and attribute-based infrastructure of federated identity management with the flexible access control of group management solutions and results in a more scalable solution. Groups can be represented as attributes and can easily be communicated in a federation.

It is in this context important to distinguish identity attributes that are used for authentication from those that are used for authorisation. Too often the attributes are mingled up and put e.g. in a single SAML authentication assertion. For fine grained access control, this approach does not work, and requires the use of more authorisation-specific solutions like XACML.

XACML increasingly becomes the de facto standard for specifying access control policies for various applications, especially web services. This dependency on XACML offers expressiveness and flexibility but also verges on the unmanageable. For federations with multiple service providers offering numerous web applications, XACML policies grow rapidly in size and complexity, which leads to longer request processing time.

Yet, heterogeneity in authorisation approaches will remain an issue. It is unrealistic to expect that every service or identity provider (or even the user) will use the same policy language for specifying their policy rules, or use the same policy decision point (PDP) for evaluating their policies. Today there are many examples of different policy languages e.g. XACML, PERMIS, Ponder etc. and many different implementations of PDPs that exist in Grid or web communities. Consequently there is a need for an authorisation infrastructure that is capable of evaluating multiple policies written in multiple languages, and supporting multiple PDPs.

What is needed is a middleware fabric that allows federation members not only to access and aggregate data from disparate data sources in a secure fashion, but also to link such sources without a reliance on a single authorisation mechanism.

### 6.1 Role of SURFnet and GigaPort3

This brings us to the roles SURFnet and/or GigaPort3 could play to further enhance and enrich authorisation functionality in the SURFFederatie and/or e-Science collaboration environments.
The hub-model of the SURFfederatie originates from the fact that authentication interoperability between IdPs and SPs had to be created. The hub facilitates this by doing protocol transformations between SAML2.0, SAML1.1, Shibboleth and A-Select. Looking at authorisation, a similar interoperability challenge is emerging. Several authorisation solutions exist that may need to be glued together at the hub. The SURFfederatie could fulfil the role of authorisation as a service provider.

**Recommendation**: Tackle heterogeneity of authorisation solutions in the SURFfederatie by letting the hub provide transformation services that enable e.g. semantic and technical interoperability. In this context, promote and stimulate the adoption of the PEP/PDP/PIP/PAP model thereby trying to create a breakthrough via ‘authorisation as a service’ offerings.

Externalising authorisation by centralising policies in a single database seems a promising solution direction. It creates a better and unified overview of the access rights to services and makes policy management more efficient. The use of the XACML framework seems suitable to achieve this ambition. It is however, unclear if there is a business case for externalising authorisation as – despite its claimed benefits – it also involves adaptation of services in order to make them XACML-enabled. Furthermore, there is little experience with XACML in the SURFfederatie.

**Recommendation**: Further investigate the possibilities and need for SURFfederatie to adopt the authorisation as a service model, i.e. study the technical possibilities and its business case. Input for the business case can be obtained by looking at existing authorisation solutions of federation members (IdPs and SPs), their maintenance costs and their willingness to adopt an authorisation as a service model.

**Recommendation**: Experiment with externalising authorisation for several SURFnet services in the SURFfederatie by centralising all service policies in a single policy. The SURFfederatie hub should host this policy and function as a PDP for the SURFnet services. The XACML policy framework should be the preferred choice.

A lot of challenges are related to the attributes that are communicated for authorisation purposes. Most of them can be solved via standardisation.

**Recommendation**: The lack of semantic interoperability between attributes that are exchanged inside and across federation should be addressed. There is a clear need of a common ontology expressing the semantics of all the elements contained in access control policies. A pragmatic approach is recommended.

In order to make fine grained access control decisions, user attributes have to be collected, stored and interpreted at different places and by different players, and they will need to be communicated and matched against each other. A framework will be required to facilitate the discovery and secure exchange of user attributes between the various involved parties.
**Recommendation:** Study approaches for attribute aggregation and communication in federated identity management. Should the SURFfederatie hub be involved with the collection of attributes, or the IdP or SP? The study should be in line with related work that is conducted in Géant3 by several other federations like SWITCH, Feide and JISC.

Delegation of authorisation introduces flexibility and scalability and features as a typical functionality in many e-Science use cases. Users should be empowered to delegate access to other users or services in friendly and intuitive manner. Yet this is not trivial as today’s solutions are based on a centralised authorisation policy system that is maintained by system administrators based on off-line communication with ‘owners’ of application or data resources. A user controlled authorisation – to authorise access to resources (data, shared workspaces etc) that are owned by a user (contrary to an institute) – such as offered by OAuth seems a better approach. OAuth allows a user to hand out tokens to third party service providers that allow them access to information stored at another service provider, without sharing their access permissions or the full extent of their data.

**Recommendation:** Research and design user centric-delegation of authorisation solutions that allow users (scientists/researchers) to delegate authorisation based on a self-service concept. Particularly, the management of OAuth delegation tokens, via concepts that are used in the User Managed Access (UMA) activity, should be further investigated.
References


[5] Progress test Feedback system, see http://prof.ivtg.nl/.


[33] OAuth, see http://oauth.net/.

[34] The OAuth Core 1.0 Protocol, draft-hammer-oauth-08, 2010, see http://tools.ietf.org/html/draft-hammer-oauth-08.


[37] SAML Profiles for OAuth, see http://liw.idcommons.net/SAML_Profiles_for_OAuth.

[38] Kantara UMA Work Group charter, see http://kantarainitiative.org/confluence/display/uma/Charter.


[40] Detailed information about UMA, see http://kantarainitiative.org/confluence/download/attachments/37751312/uma-webinar-2010-01-29.pdf.

[41] UMA deep dive slide deck, see http://kantarainitiative.org/confluence/download/attachments/37751312/UMA-protocol-flow-deep-dive.pdf.


Authorisations and Group Management


[53] Privilege and Access Management working group of the Middleware Architecture Committee for Education (MACE-paccman), see http://middleWare.internet2.edu/paccman/.


[58] CAS is not trivial to set up, see www.nesc.ac.uk/hub/projects/etf.


[64] SWITCH VO concept, see http://www.switch.ch/aai/about/vo-concept/.


[67] Open Liberty OpenAz project, see http://www.openliberty.org/wiki/index.php/Main_Page#OpenAz.


[81] Ivonne Thomas & Christoph Meinel, An Identity Provider to manage Reliable Digital Identities for SOA and the Web, IDtrust ‘10, April 13-15, 2010, Gaithersburg, MD.
[83] USGov/Internet2 Tao of Attributes workshop 2009, see http://middleware.internet2.edu/tao-of-attributes/.
[84] Kantara Initiative website, see http://kantarainitiative.org/.