# PERMIS: A Modular Authorization Infrastructure

David Chadwick, Gansen Zhao, Sassa Otenko, Romain Laborde, Linying Su, Tuan Anh Nguyen

University of Kent

**Abstract**

Authorization infrastructures manage privileges and render access control decisions, allowing applications to adjust their behavior according to the privileges allocated to users. This paper describes the PERMIS role based authorization infrastructure along with its conceptual authorization, access control, and trust models. PERMIS has the novel concept of a credential validation service, which verifies a user's credentials prior to access control decision making and enables the distributed management of credentials. PERMIS also supports delegation of authority, thus credentials can be delegated between users, further decentralizing credential management. Finally, PERMIS supports history based decision making which can be used to enforce such things as separation of duties and cumulative use of resources. Details of the design and the implementation of PERMIS are presented along with details of its integration with Globus Toolkit, Shibboleth and GridShib. A comparison of PERMIS with other authorization and access control implementations is given, along with suggestions where future research and development is still needed.

**Keywords: PDP, authorization infrastructure, access control decisions, grid security**

## 1. Introduction

Policy based authorization infrastructures contain a number of advantages over access control lists and hard coded systems. They are more flexible and scalable, and are application independent. They provide facilities to manage user privileges, render access control decisions, and process the related information. Different types of policies may be supported, such as Credential Issuing Policies, Access Control Policies, Delegation Policies, and Credential Validation Policies. These policies contain the rules and criteria that specify how user privileges (or credentials, which are digitally signed assertions made by some authority about a user's privileges) are managed and access control decisions are made. In the context of distributed grid systems spanning multiple domains, policy based authorization systems bring a number of specific advantages such as: they can control the issuing of credentials in one domain and allow the autonomous delegation of privileges between users. They can then separately control the validation of these credentials in the resource domain, and allow each resource owner to independently say who he trusts to issue which credentials to whom, and which access rights these valid credentials should have. This is an important feature that most grid systems today do not have.

The authorization infrastructure that we have built is called PERMIS [1]. This paper describes the various components of the PERMIS authorization infrastructure, the conceptual models that lie behind them, and the standards that we have used. We conclude by comparing our work to that of others and describing some of the future work that still needs to be done. The rest of this paper is structured as follows. Section 2 provides the conceptual models of our authorization infrastructure. Section 3 describes the design and implementation of PERMIS. Section 4 presents PERMIS's integration with Globus Toolkit, Shibboleth and GridShib. Section 5 compares PERMIS to other related research. Section 6 concludes and indicates our plans for the future.

## 2. Conceptual Models

### 2.1 The Access Control and Authorization Models

The authorization model paradigm that we have adopted is the well known "Subject – Action – Target" paradigm combined with an enhancement of the ISO Attribute Based Access Control (ABAC) model [25]. Because grids are distributed systems we cannot assume that all the attributes claimed by a user are rightfully his. Consequently we have enhanced ABAC so that subject attributes are presented as digitally signed credentials issued to the subject by one or more trusted (in the eyes of the resource owner) attribute authorities (AAs). A Credential Validation Service is introduced to validate these credentials and determine which of the attributes can rightfully be claimed by the subject. Each resource owner specifies the credential validation policies for gaining access to his resources.

ABAC is a generalization of the well known role based access control (RBAC) model [18], in which a role is not restricted to an organizational role, but can be any attribute of the subject, such as a professional qualification or their current level of authentication (LoA) [23]. In the following discussion we will refer to roles, on the assumption that we mean any attribute that can assigned to a subject. Each subject represents a real world principal, which is the action performer. Subjects are often referred to as users, but they are not limited to human beings. Action is the operation that is requested to be performed on the target. It can be either a simple operation, or a bundle of complex operations that is provided as an integrated set. Target is the object of the action, over which the action is to be performed. A target represents one or more critical resources that need to be protected from unauthorized access.

PERMIS uses the RBAC (or ABAC) model, in which roles are used to model organization roles, user groups, or any attribute of the user. Subjects are assigned attributes, or role memberships. A subject can be the member of zero, one or multiple roles at the same time. Conversely, a role can have zero, one or more subject occupants at the same time. In RBAC a role is associated with a set of privileges, where each privilege is the right to perform a particular action on a particular target. The PERMIS model is more flexible and allows sets of privileges to be assigned to sets of roles, rather than to single roles, since the latter is too restrictive in practice. For example if project managers have some organizational based privileges, project members have some project specific privileges, and project managers have some higher level project specific privileges, then without the ability to assign the latter to a combination of roles (project member and project manager), a new set of roles have to be specially created for each project manager. Thus each subject is authorised to perform the actions corresponding to his role memberships. Changing the privileges allocated to a set of roles will affect all subjects who are members of the role set (or who have been assigned the set of attributes).

PERMIS supports hierarchical RBAC in which roles (or attributes) may be organized in a partial hierarchy, with some being superior to others. A superior role inherits all the privileges allocated to its subordinate roles. For example, if the role Staff is subordinate to Manager, then the Manager role will inherit the privileges allocated to the Staff role. A member of the Manager role can perform operations explicitly authorized to Managers as well as operations authorised to Staff. The inheritance of privileges from subordinate roles is recursive, thus a role $r_o$ will inherit privileges from all its direct subordinate roles $r_s$, and indirect subordinate roles which are direct or indirect subordinate roles of $r_s$. Role hierarchies need not apply only to organizational roles, but can apply to any attribute, such as level of authentication (LoA), where there is a natural precedence in the attribute values, in which a higher value implies the privileges of the lower values. In the LoA case, a user who has been authenticated to LoA value 4 (the highest) can be assumed to inherit the privileges assigned to the lower levels of authentication.

Figure 1 shows our high level conceptual model for an authorization infrastructure. Step 0 is the initialization step for the infrastructure, when the policies are created and stored in the various components. Each subject may possess a set of credentials from many different Attribute

Authorities (AAs), that may be pre-issued, long lived and stored in a repository or short lived and

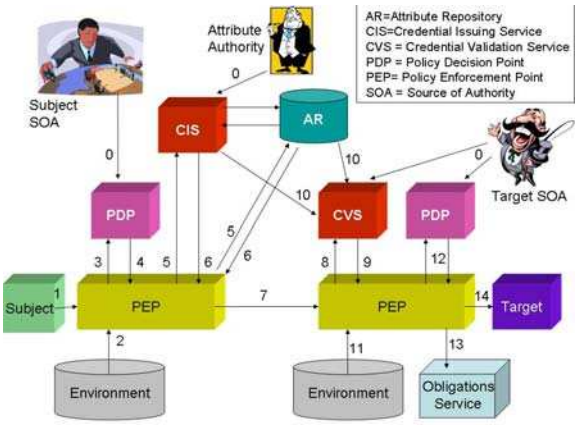


**Figure 1: High Level Conceptual Model of an Authorization Infrastructure**

issued on demand, according to their Credential Issuing Policies. The Subject Source of Authority (SOA) dictates which of these credentials can leave the subject domain for each target domain. When a subject issues an application request (step 1), the application independent policy decision point (PDP) informs the application's policy enforcement point (PEP) which credentials to include with the user's request (steps 3-4). These are then collected from the Credential Issuing Service (CIS) or Attribute Repository by the PEP (steps 5-6). The user's request is transferred to the target site (step 7) where the target SOA has already initialized the Credential Validation Policy that says which credentials from which issuing AAs are trusted by the target site, and the Access Control policy that says which privileges are given to which attributes. The user's credentials are first validated (step 8). This may require the CVS to pull additional credentials from an AA's repository or issuing service (step 10). The valid attributes are returned to the PEP (step 9), combined with any environmental information, such as current date and time (step 11), and then passed to the PDP for an access control decision (step 12). If the decision is granted the user's request is allowed by the PEP (step 14), otherwise it is rejected. In either case, the PDP may also return a set of obligations, which are actions that the PEP must enforce along with the access control decision (step 13). An obligations service is the functional component that is responsible for enacting these obligations. In more sophisticated systems there may be a chain of PDPs that are called by a master PDP, with each PDP in the chain holding a different policy possibly written by a different SOA and possibly written in a different policy language. In this case the master PDP needs to hold a policy combining policy written by the target SOA, which determines the ultimate response to give to the PEP based on the set of granted, denied or don't know responses returned by the chain of PDPs. Application PEPs however should be shielded from needing to know about this more sophisticated functionality.

### 2.2 The Trust and Delegation Models

Credentials are the format used to securely transfer a subject's attributes/roles from the Attribute Authority to the recipient. They are also known as attribute assertions [20]. PERMIS only trusts valid credentials. A valid credential is one that has been issued by a trusted AA or his delegate in accordance with the current authorization policies (Issuing, Validation and Delegation policies).

It is important to recognize the difference between an authentic credential and a valid credential. An authentic credential is one that has been received exactly as it was originally issued by the AA. It has not been tampered with or modified. Its digital signature, if present, is intact and validates as trustworthy by the underlying PKI, meaning that the AA's signing key has not been compromised, i.e. his public key (certificate) is still valid. A valid credential on the other hand is an authentic credential that has been issued according to the prevailing authorization policies. Credential authenticity is a concern of the authentication system whilst credential validity is a concern of the authorization system. In order to clarify the difference, an example is the paper money issued by the makers of the game Monopoly. This money is authentic, since it has been issued by the makers of Monopoly. The money is also valid for buying houses on Mayfair in the game of Monopoly. However, the money is not valid if taken to the local supermarket

because their policy does not recognize the makers of Monopoly as a trusted AA for issuing money. Nevertheless, the money still remains authentic. This is a real problem in the context of grids today. VOMS servers [6] issue credentials and sign them with public key certificates issued by trusted grid CAs, therefore the credentials they issue are authentic. However without a proper functioning authorization system, a grid resource cannot tell the difference between a VOMS credential issued by a VOMS server managed by a trustworthy organization and one that has been quickly set up by a student who has a valid grid public key certificate, since both sets of VOMS issued credentials are authentic.

Recognition of trusted AAs is part of PERMIS's Credential Validation Policy. The Credential Validation Service (CVS) is the component that checks that each credential issuer is mentioned in this policy directly, or that the credential issuer has been delegated a privilege by a trusted AA either directly or indirectly (i.e. a recursive chain of trusted issuers is dynamically established controlled by the Delegation Policies of the Target SOA and the intermediate AAs in the chain). The PERMIS Credential Validation Policy contains rules that govern which attributes different AAs are trusted to issue, along with a Delegation Policy for each AA. These rules separate AAs into different groups and assign them different rights to issue different attributes to different sets of subjects. Further each AA will have its own Credential Issuing Policy and Delegation Policy. PERMIS assumes that if a credential has been issued and signed by a trusted AA, then it must be conformant to the AA's Issuing Policy, so this need not be checked any further. However, if the credential was subsequently delegated this may or may not have conformed to the original AA's Delegation Policy. Therefore when the CVS validates a delegated credential it needs to check that it conforms to the AA's delegation policy as well as the Target SOA's delegation policy. This can only be done if the AA makes its delegation policy available to the CVS, which typically means that it must insert its policy into each issued credential. Current international standards for the format of credentials only have limited support for this feature at the moment. For example, X.509 attribute certificates [3] may contain a path length constraint which can be set by an AA to limit the length of the delegation chain, and a name constraints that limits who the delegates can be. As international standards add more delegation policy fields to their credential formats, then the PERMIS CVS will be able to validate that more of the AA's delegation policy has been adhered to.

The current PERMIS delegation model constrains delegations to a tree rather than a directed graph, since this simplifies the process of credential validation and credential revocation. A delegate can be given a privilege to either delegate to others or assert or both. Each AA may further constrain delegations by validity times and delegation chain lengths. PERMIS also ensures that all delegated credentials conform to the following delegation paradigms:

i) an issuer cannot delegate more privileges than he possesses, to ensure constrained propagation of privileges from issuers to subjects, and

ii) an issuer cannot delegate a privilege to himself or to a superior in the delegation chain, since the recipient already holds this privilege. The only reason an issuer may want to do this, would be to circumvent the control that he is allowed to delegate but not assert a privilege, and by delegating to himself or to a superior he would be allowed to remove this control.

The net result of this trust model is that PERMIS can support multiple AAs issuing different sets of attributes to the same or different groups of users, in which each AA can have different delegation policies, yet the target SOA can specify an overall Credential Validation Policy that constrains which of these (delegated) credentials are trusted to be used to access the resources under his control. Originally the model assumed that each subject would be known by the same globally unique name (typically an X.500 distinguished name) at each AA. We now know this isn't always the case, and so this may be addressed by providing a name mapping function that can map between the different

names of a subject in different issuing domains. This is the approach that is currently being adopted in projects such as GridShib [10] and Shintau [26].

### 2.3 The Coordinated Decision Making Model

Sometimes coordination is needed between access control decisions. For example, in order to support mutually exclusive tasks (Separation of Duties), the PDP needs to know if the same user is trying to perform a second task in a set of mutually exclusive ones. Alternatively, if multiple resources are available but their use is to be restricted, for example, a maximum of 30GB of storage can be used throughout a grid, then each PDP needs to know what the other PDPs have already granted the user access to. One model is to use a stateful PDP, which retains information about previous access control decisions, in so called *retained ADI* [25]. This allows coordination between successive access control decisions in the same PDP. Extending this model and communicating the *retained ADI* between a set of stateful PDPs would allow coordinated access control decisions to be made by multiple distributed PDPs. However, most PDPs that have been built today are not stateful, nor do they have the ability to communicate with each other. Consequently an alternative model is to store the *retained ADI* in a central secure database that is accessible (indirectly) by all the PDPs, and that can be updated (indirectly) by them. In this model the *retained ADI* is modeled as attributes of a Coordination object, and a Policy Information Point (PIP) retrieves this information on behalf of the PDPs. Obligations in the policy say how this information should be updated, and the Obligations service performs the updates on behalf of the PDPs. A fuller description of this model and its implementation in PERMIS can be found in [29]. Modifying PERMIS to hold retained ADI and support Separation of Duties policies is described in [28].

## 3. PERMIS: A Modular Authorization Infrastructure

The PERMIS authorization infrastructure is shown in Figure 2. The PERMIS authorization infrastructure provides facilities for policy management, credential management, credential validation and access control decision making. It is necessary for applications to intercept users' requests, ask PERMIS to validate the user's

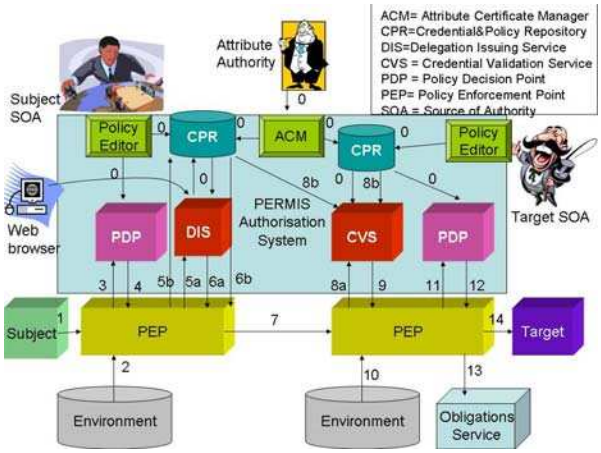


**Figure 2: Architecture of the PERMIS Authorization Infrastructure**

credentials and make an access control decision, and then enforce the access control decisions and obligations that are returned by PERMIS.

### 3.1 Policy Management

PERMIS Policies are rules and criteria that the decision making process uses to render decisions. It mainly contains two categories of rules, trust related rules (Credential Validation Policy) and privilege related rules (Access Control Policy). Trust related rules specify the system's trust in the distributed Attribute Authorities, and which attributes they are allowed to assign to whom. Only credentials issued by trusted AAs within their authority will be accepted. Privilege related rules specify the domains of targets, the actions supported by the targets, the role hierarchies, the privileges assigned to each role and the conditions under which these privileges may be used, for example, the times of day or the maximum amount of a resource that may be requested. In terms of the RBAC model, the trust related rules control the user-role assignments, whilst the privilege related rules control the role-privilege assignments.

PERMIS provides a policy management tool, the Policy Editor [13] (see Figure 3), which users can use to compose and edit PERMIS policies. The GUI interface of the Policy Editor

comprises: the subject policy window (Where Users Are From), the trusted AA policy window (User Account Administators), the user-role assignment policy window (Account Administrator Privileges), the role hierarchy policy window (User's Roles), the target resource policy window (My Protected Resources), the action policy window (Resources' Functions) and the role-privilege

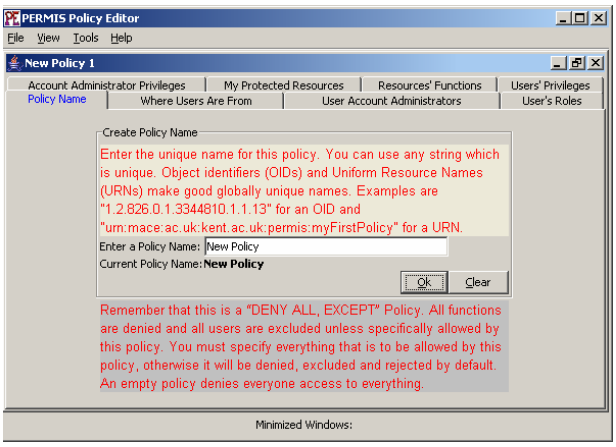


**Figure 3. The PERMIS Policy Editor**

assignment policy window (Users' Privileges). These windows provide forms for users to fill in, then the tool generates the corresponding PERMIS policy in XML. Policies can be saved as pure XML in text files, or the XML can be embedded as a policy attribute in an X.509 Attribute Certificate (AC) [3], digitally signed with the policy author's private key (held in a PKCS#12 file) then stored in either a local file, LDAP directory or WebDAV [27] repository. Various helpers in the Policy Editor are capable of retrieving subject and AA names from LDAP directories, and setting times and dates in the correct format. Authors can use the Policy Editor to browse the LDAP directories and WebDAV repositories to select existing policies to update them.

### 3.2 Credential Management

The Credential Management system is responsible for issuing and revoking subject credentials. The Attribute Certificate Manager (ACM) tool is used by administrators to allocate attributes to users in the form of X.509 ACs. These bind the issued attributes with the subject's and issuer's identities in a tamper-proof manner. The ACM has a GUI interface that guides the manager through the process of AC creation, modification and revocation. The manager can search for a user in an attached LDAP directory or WebDAV repository, or enter the DN of the user directly. There is then a picking list of attribute types (e.g. role, affiliation etc.), to which the manager can add his own value (e.g. project manager). There is a pop up calendar allowing the manager to select the dates between which the AC is valid, plus the option of adding appropriate times of day to these. Finally the manager can add a few standard selected extensions to the AC, to say whether the holder is allowed to further delegate or not, and if so, how long the delegation chain can be ("basic attribute constraints" extension [3]), or if the holder may assert the attributes or only delegate them to others ("no assertion" extension [4]). Finally, the manager must add his digital signature to the AC, so the GUI prompts him for the PKCS#12 file holding his private key and his password to unlock it. Once the AC is signed, the manager has the option of storing it in an LDAP directory, WebDAV repository or local filestore. Besides creating ACs, the ACM allows the manager to edit existing ACs and to revoke existing ACs by deleting them from their storage location. Note that at present revocation lists have not been implemented, because short validity times or deletion from storage have been sufficient to satisfy our current user requirements.

The Delegation Issuing Service (DIS) is a web service that dynamically issues X.509 ACs on demand when requested to by the delegator. It may be called directly by an application's PEP after a user has invoked the application, to issue short lived ACs to the application for the duration of the user's task. Alternatively there is a http interface that lets users invoke it via their web browers to dynamically delegate their existing longer lived credentials to other users, so as to enable them to act on their behalf. This is especially powerful, as it empowers users to delegate (a subset of) their privileges to other users without any administrative involvement. Because the DIS is controlled by its own

PERMIS policy, written by the Subject SOA, an organization can tightly control who is allowed to delegate what to whom, and then leave its subjects to delegate as they see fit. The DIS stores all delegated credentials in a locally configured LDAP server or WebDAV repository, so that they can be retrieved on demand by the authorization system (in steps 6b and 8b). The DIS has a number of advantages over the ACM, such as: users do not need to have X.509 public key certificates as all issued credentials are signed by the DIS, delegation chains are kept to a maximum length of 2, and revoking a user's credential does not automatically revoke any credentials he may have already delegated. More details of the DIS can be found in [2].

### 3.3 Authorization Decision Engine

The PERMIS Authorization Decision Engine is responsible for credential validation and access control decision making. Credential validation is the process that enforces the trust and delegation model of PERMIS as described in Section 2.2, and ensures that only valid roles/attributes are attributed to users. Access control decision making is the process that ensures only users with the required attributes gain access to the protected resources. Together they enforce the enhanced ABAC model described in Section 2.1. The CVS extracts the subset of valid attributes from the set of available credentials, according to the Target SOA's Credential Validation Policy. The PDP makes access control decisions based on the Target SOA's access control policy and the valid attributes passed from the CVS. The PERMIS authorization decision engine is superior to conventional PDPs since it has the ability to validate credentials and delegation chains, which is not a common capability of conventional PDPs e.g. Sun's XACML PDP [15]. Furthermore it supports history based decision making and multi-session separation of duties [28]. Figure 4 depicts the overall architecture of the PERMIS Authorization Decision Engine. It comprises five main components: the PDP, the CVS, the Credential Retriever, the Credential Decoder, and the Policy Parser.

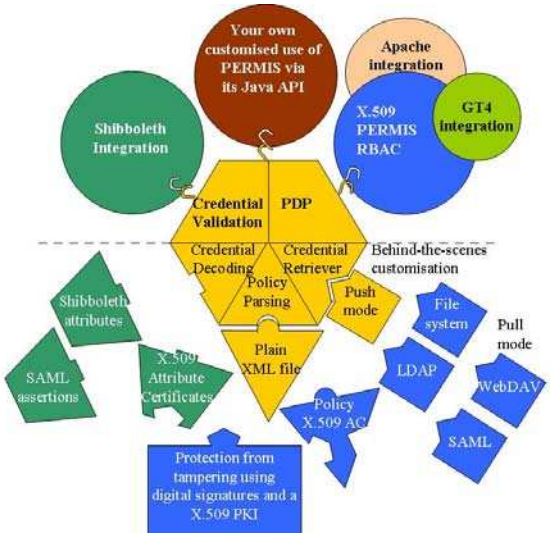


**Figure 4: The PERMIS Authorization Decision Engine**

### 3.4 The PDP

The PDP component is responsible for making access control decisions based on the valid attributes of the user and the Target SOA's access control policy, which is a subset of the PERMIS policy.

At initialization time the Target SOA's PERMIS policy is read in (step 0 in Figure 2) and parsed by the Policy Parser so that both the PDP and CVS are ready to operate. Both plain XML policies and digitally signed and protected policies can be read in. The former are stored as text files in the local filestore whilst the latter are stored as X.509 policy ACs in either the local filestore, or the Target SOA's entry in an LDAP directory or WebDAV repository. X.509 ACs are tamper resistant and integrity protected, whereas text files have to be protected by the operating system.

Each time the user makes a request to the application to perform a task (step 1 or 7 in Figure 2), the PEP passes this request to the PERMIS PDP (step 3 or 11) along with user's valid attributes and any required environmental attributes such as the time of day. The PEP needs to know which environmental attributes are needed by the access control policy, and since the PEP is application specific software, it is more likely that the access control policies will be restricted to constraints based on the

environmental attributes that the PEP is capable of passing to the PDP.

### 3.5 The CVS

As described in Section 2.2, all credentials allocated to subjects will be validated by the CVS according to the Target SOA's credential validation policy, which is a subset of the PERMIS policy. Figure 5 illustrates the detailed architecture of the CVS, along with the internal data flows and sequence of events.

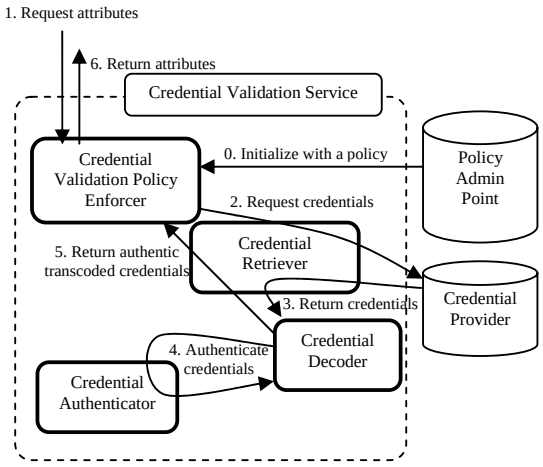


**Figure 5: Data Flow Diagram for Credential Validation Service Architecture**

First of all the service is initialised by giving it the credential validation policy. The policy parsing module described in Section 3.4 is responsible for this. When the user activates the application, the target PEP requests the valid attributes of the subject (step 1 in Fig 4, step 8a in Fig 2). Between the request for attributes and returning them (in step 6 or 9 respectively) the following events may occur a number of times i.e. the CVS is capable of recursively calling itself as it determines the path in a delegation tree from a given credential to a trusted AA specified in the policy.

The Credential Validation Policy Enforcer requests credentials from the Credential Retriever (step 2). PERMIS can operate in either credential pull mode or credential push mode. In credential push mode the application passes the user's credentials along with his request to the target PEP (Step 7 in Fig 2) and the PEP passes them to the CVS. In credential pull mode, the credentials are dynamically pulled from one or more remote credential providers (these could be AA servers, LDAP or WebDAV repositories etc.) by the CVS (step 8b in Fig 2, step 2 in Fig 4). The actual attribute request protocol (e.g. SAML or LDAP) is handled by the appropriate Credential Retriever module, whilst the credential format is handled by the appropriate Credential Decoder module. When operating in credential push mode, the PEP stores the already obtained credentials in a local Credential Provider repository and pushes the repository to the CVS, so that the CVS can operate in logically the same way for both push and pull modes. After credential retrieval, the credentials are passed to the Credential Decoding module (step 3 Fig 4). From here they undergo the first stage of validation – credential authentication (step 4). Because only the Credential Decoder is aware of the actual format of the credentials, it has to be responsible for authenticating the credentials using an appropriate Credential Authenticator module. Consequently, both the Credential Decoder and Credential Authenticator modules are encoding specific modules. For example, if the credentials are digitally signed X.509 ACs, the Credential Authenticator uses the configured X.509 PKI to validate the signatures. If the credentials are XML signed SAML attribute assertions, then the Credential Authenticator uses the public key in the SAML assertion to validate the signature. The Credential Decoder subsequently discards all unauthentic credentials – these are ones whose digital signatures are invalid. Authentic credentials are decoded and transformed into an implementation specific local format that the Policy Enforcer is able to handle (step 5).

The task of the Policy Enforcer is to decide if each authentic credential is valid (i.e. trusted) or not. It does this by referring to the Credential Validation Policy to see if the credential has been issued by a trusted AA or not. If it has, it is valid. If it has not and it is a delegated credential, the Policy Enforcer has to work its way up the delegation tree from the current credential to its issuer and from there to its issuer, recursively, until a trusted AA is located, or no further issuers can be found (in which case the credential is not trusted and is discarded). Consequently steps 2-5

are recursively repeated until closure is reached (which, in the case of a loop in the credential chain, will be if the same credential is encountered again). Remember that in the general case there are multiple trusted credential issuers, who each may have their own Delegation Policies, and these must be enforced by the Policy Enforcer as much as is possible from what has been provided in the issued credentials.

The CVS can be customized by PERMIS implementers, by implementing their own credential retrieval and decoding services and plugging them into PERMIS. This enables implementers to adopt credential formats and retrieval protocols that are not yet implemented by PERMIS, such as local proprietary formats. PERMIS can theoretically be customized to support any application specific credential validation requirements.

## 4. Integrating PERMIS

### 4.1 Integration with GT4

Globus Toolkit (GT) is an implementation of Grid software, which has a number of tools that make development and deployment of Grid Services easier [9]. One of the key features of this toolkit is secure communications. However, Globus Toolkit has limited authorization capabilities based on simple access control lists and grid mapfiles. To improve its authorization capabilities a Security Assertions Markup Language (SAML) authorization callout has been added. SAML [20] is a standard designed by the Organization for the Advancement of Structured Information Standards (OASIS) to provide a universal mechanism for conveying security related information between the various parts of an access control system. The Open Grid Forum (OGF) has produced a profile of SAML for use in Grid authorization [19]. Consequently it is now possible to deploy an external authorization service that GT will contact to make authorization decisions on its behalf. A standalone PERMIS Authorization Service has been developed to provide this type of authorization decision to GT3 and GT4 through the SAML callout [8].

PERMIS has also been integrated with GT4 via its Java call outs to custom Policy Information Points (PIPs) and PDPs. In this case the PERMIS CVS is configured as a custom PIP and the PERMIS PDP is configured as a custom PDP. Information between the two modules is carried in the format of an XACML request context [14].

### 4.2 Integration with Shibboleth

Shibboleth [21] is a cross-institutional authentication and authorization architecture for single sign on and access control of web resources. Shibboleth defines a protocol for carrying authentication information and user attributes from the user's home site to the resource site. The resource site can then use the user attributes to make an access control decision about the user's request. A user only needs to be authenticated once by the home site in order to visit other Shibboleth protected resource sites in the federation, as the resulting authentication token is recognized by any member of the federation. In addition to this, protection of the user's privacy can be achieved, since the user is able to restrict what attributes will be released to the resource providers from his/her home site. However Shibboleth's built in access control decision making based on the user's attributes is simplistic in its functionality, and the management of the access controls is performed together with web server administration at the resource site. Furthermore, distributed management of credentials and dynamic delegation of authority are not supported. To rectify these deficiencies, a Shibboleth-Apache Authorization Module (SAAM) has been developed which integrates PERMIS with Shibboleth. SAAM plugs into Apache and replaces the Shibboleth authorization functionality with calls to the PERMIS authorization decision engine. A full description is provided in [5]

PERMIS extends the access control model used in Shibboleth by introducing hierarchies of roles, distributed management of attributes, and policy controlled decisions based on dynamically evaluated conditions. PERMIS supports the existing semantics of Shibboleth attributes, but also allows X.509 ACs to be used instead, where more secure credentials are needed.

### 4.3 Integration with GridShib

GridShib [10] provides interoperability between Globus Toolkit [9] and Shibboleth [21]. The GridShib Policy Information Point (PIP) (see Figure 6) retrieves a user's attributes from the Shibboleth Identity Provider (IdP). The Distinguished Name Binder component is responsible for mapping the user's DN, obtained by the GridShib PIP from the proxy certificate, into the user's Shibboleth identity. The retrieved attributes are parsed and passed to the GT4 PEP which then feeds them to the PDP for an authorization decision. GridShib integrates Shibboleth's attribute management functionality with GT4's authorization decision making for Grid jobs. However, like GT4, GridShib provides only limited PDP functionality, which is based on access control lists and is not capable of coping with dynamically changing conditions, which a policy based engine is.

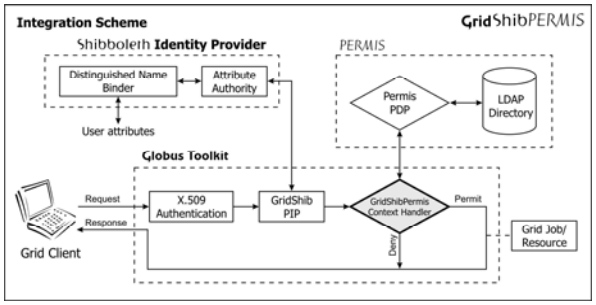


**Figure 6: GridShibPERMIS Integration Scheme**

GridShibPERMIS provides a GridShibPERMIS Context Handler that can be integrated with GT4 as a Java callable PDP. The Context Handler is invoked by GT4 when an authorization decision is to be made. The Context Handler is fed with the user's attributes that have been retrieved from the Shibboleth IdP. They are parsed and stored in a local Credential Provider Repository, ready to be accessed by the PERMIS CVS as described in Section 3.5. The Context Handler calls the CVS, which ensures that the attributes are valid according to the Target SOA's policy; then calls the PDP, which renders an access control decision, and finally it returns the result to GT4.

## 5. Related Work

Manandhar et at. [12] present an application infrastructure in which a data portal allows users to discover and access data over Grid systems. They propose an authorization framework that allows the data portal to act as a proxy and exercise the user's privileges. When a user authenticates to the data portal, a credential is generated stating that the data portal is authorized to exercise the user's privileges for a specific period. The credential is then used by the data portal to retrieve the user's authorization tokens from various providers. When requesting a service from a service provider, the data portal presents both the credential and the authorization tokens. The authorization decision is then made by the service provider. The proposed infrastructure mainly focuses on the interaction between different systems in the Grid environment, with no in depth discussion about the access control model or the trust model. Credential verification is also missing from the discussion.

XACML [14] defines a standard for expressing access control policies, authorization requests, and authorization responses in XML format. The policy language allows users to define application specific data types, functions, and combining logic algorithms, for the purpose of constructing complex policies. Sun's open source XACML implementation [15] is a java implementation of the XACML 2.0 standard and provides most of the features in the standard. The XACML policy language is richer than that of PERMIS's PDP policy, but XACML has not yet addressed the issue of credential validation and is only now working on dynamic delegation of authority [22].

The Community Authorization Service (CAS) [11] was developed by the Globus team to improve the manageability of user authorization. CAS allows a resource owner to grant access to a portion of his/her resource to a VO (or community – hence the name CAS), and then let the community determine who can use this allocation. The resource owner thus partially delegates the allocation of authorization rights to the community. This is achieved by having a CAS server, which acts as a trusted intermediary between VO users and resources. Users first contact the CAS asking for permission to use a

Grid resource. The CAS consults its policy (which specifies who has permission to do what on which resources) and if granted, returns a digitally self-signed capability to the user optionally containing policy details about what the user is allowed to do (as an opaque string). The user then contacts the resource and presents this capability. The resource checks that the capability is signed by a known and trusted CAS and if so maps the CAS's distinguished name into a local user account name via the Grid mapfile. Consequently the Grid mapfile now only needs to contain the name of the trusted CAS servers and not all the VO users. This substantially reduces the work of the resource administrator. Further, determining who should be granted capabilities by the CAS server is the task of other managers in the VO community, so this again relieves the burden on resource managers. For finer grained access control, the resource can additionally call a further routine, passing to it the opaque policy string from the capability, and using the returned value to refine the access rights of the user. Unfortunately this part of the CAS implementation (policy definition and evaluation routine) were never fully explored and developed by the Globus team. This is precisely the functionality that PERMIS has addressed.

The main purpose of SPKI [16] is to provide public key infrastructures based on digital certificates without depending upon global naming authorities. SPKI binds local names and authorizations to public keys (or the hash values of public keys). Names are allocated locally by certificate issuers, and are only of meaning to them. SPKI allows authorizations to be bound directly to public keys, removing the process of mapping from authorization to names and then to public keys. SPKI supports dynamic delegation of authorizations between key holders, and allocation of authorizations to groups. Though SPKI can convey authorization information, it does not cover authorization decision making or access control policy issues. One can thus regards SPKI as an alternative format to X.509 ACs or SAML attribute assertions for carrying credentials, and PERMIS could easily be enhanced to support this format of credential if it were required.

The EU DataGrid and DataTAG projects have developed the Virtual Organisation Membership Service (VOMS) [6] as a way of delegating the authorization of users to managers in the VO. VOMS is a credential push system in which the VOMS server digitally signs a short lived X.509 role AC for the VO user to embed in his proxy certificate and present to the resource. The AC contains role and group membership details, and the Local Centre Authorization Service (LCAS) [7] makes its authorization decision based upon the user's AC and the job specification, which is written in job description language (JDL) format. This design is similar in concept to the CAS, but differs in message format and syntax. However what neither VOMS nor CAS nor LCAS provide is the ability for the resource administrator to set the policy for access to his/her resource and then let the authorization infrastructure enforce this policy on his/her behalf. This is what systems such as PERMIS and Keynote [17] provide. It will therefore be relatively easy to replace LCAS with the PERMIS decision engine, so that VOMS allocated role ACs can be pushed to the resource site for PERMIS to make the policy controlled authorization decisions. This is the subject of the current VPMan project [24].

KeyNote [17] is a trust management system that provides a general-purpose mechanism for defining security policies and credentials, and rendering authorization decisions based on them. KeyNote provides a language for defining both policies and assertions, where policies state the rules for security control, and assertions contain predicates that specify the granted privileges of users. KeyNote has been implemented and released as an open source toolkit. But KeyNote is not without its limitations. Keynotes policies and credentials are in their own proprietary format. KeyNote credentials have no time limit, and Keynote has no concept of revocation of credentials. Further, policies define the roots of trust, but the policies themselves are not signed and therefore have to be stored securely and are only locally trusted.

## 6. Conclusions

This paper briefly presents our work on designing and building a modular policy based

authorization infrastructure. We have explained the conceptual models that underpin PERMIS and summarized the design and the implementation of the various functional components that comprise the PERMIS authorization infrastructure. These provide support for policy management, attribute management, and authorization decision making. We have provided details about our new conceptual component, the credential validation service. Finally, we have presented a comparison of related work, pointing out their relative advantages and disadvantages as compared to PERMIS.

### 6.1 Future Work

Obligations are actions that are required to be fulfilled along with the enforcement of access control decisions. Whilst PERMIS already supports obligations in its policies and will return them along with its access control decisions, the major area for research and development is building a general purpose application independent Obligations Service to form part of the authorization infrastructure.

Constructing a master PDP that can coordinate the calling of multiple subordinate PDPs and can combine their varying decisions into one overall decision for the PEP, is another area for research and development.

Finally, defining a standard mechanism for aggregating attributes from multiple authorities, where the user is known by different names at the different authorities, is another challenging avenue of research which we are currently undertaking in the Shintau project [26].

## Acknowlegements

We would like to thank the UK JISC for funding part of this work under the DyCOM, DyVOSE, SIPS and GridAPI projects, and the EC for funding part of this work under the TrustCoM project (FP6 project number 001945).

## References

1. D.W.Chadwick, A. Otenko "The PERMIS X.509 Role Based Privilege Management Infrastructure". Future Generation Computer Systems, 936 (2002) 1–13, December 2002. Elsevier Science BV.

2. D.W.Chadwick. "Delegation Issuing Service". NIST 4th Annual PKI Workshop, Gaithersberg, USA, April 19-21 2005

3. ISO 9594-8/ITU-T Rec. X.509 (2001) "The Directory: Public-key and attribute certificate frameworks"

4. ISO 9594-8/ITU-T Rec. X.509 (2005) "The Directory: Public-key and attribute certificate frameworks"

5. Wensheng Xu, David Chadwick, Sassa Otenko. "Development of a Flexible PERMIS Authorization Module for Shibboleth and Apache Server". Proceedings of 2nd EuroPKI Workshop, University of Kent, July 2005

6. R. Alfieri et al. "VOMS: an Authorization System for Virtual Organizations", 1st European Across Grids Conference, Santiago de Compostela, February 13-14, 2003

7. Martijn Steenbakkers "Guide to LCAS v.1.1.16", Sept 2003. Available from http://www.dutchgrid.nl/DataGrid/wp4/lcas/edg-lcas-1.1

8. David Chadwick, Sassa Otenko, and Von Welch. "Using SAML to Link the GLOBUS Toolkit to the PERMIS Authorization Infrastructure". In Proceedings of Eighth Annual IFIP TC-6 TC-11 Conference on Communications and Multimedia Security, Windermere, UK, September 2004.

9. I. Foster. "Globus Toolkit Version 4: Software for Service-Oriented Systems". IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2005.

10. Barton, T., Basney, J., Freeman, T., Scavo, T., Siebenlist, F., Welch, V., Ananthakrishnan, R., Baker, B., and Keahey, K. "Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, Gridshib, and MyProxy", 5th Annual PKI R&D Workshop. April 2006.

11. Ian Foster, Carl Kesselman, Laura Pearlman, Steven Tuecke, and Von Welch. "The Community Authorization Service: Status and Future". In Proceedings of Computing in High Energy Physics 03 (CHEP '03), 2003.

12. Ananta Manandhar, Glen Drinkwater, Richard Tyer, Kerstin Kleese. "GRID Authorization Framework for CCLRC Data Portal", Second Earth Science Portal Workshop: Web Portal Framework Design/Implementation, September 2003.

13. Sacha Brostoff, M. Angela Sasse, David Chadwick, James Cunningham, Uche Mbanaso, Sassa Otenko. "*"R-What?"* Development of a Role-Based Access Control (RBAC) Policy-Writing Tool for e-Scientists" Software: Practice and Experience Volume 35, Issue 9, Date: 25 July 2005, Pages: 835-856

14. OASIS. "XACML 2.0 Core: eXtensible Access Control Markup Language (XACML) Version 2.0", Oct, 2005.

15. Sun's XACML Implementation available on http://sunxacml.sourceforge.net/.

16. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomsa, and T. Ylonen. "SPKI Certificate Theory". RFC 2693, September 1999.

17. M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. "The KeyNote Trust Management System Version 2". RFC 2704, Sept. 1999.

18. David F. Ferraiolo and Ravi Sandhu and Serban Gavrila and D. Richard Kuhn and Ramaswamy Chandramouli. “Proposed NIST standard for role-based access control”. ACM Transactions on Information and System Security Volume 4, Issue 3. August 2001.
19. Von Welch, Rachana Ananthakrishnan, Frank Siebenlist, David Chadwick, Sam Meder, Laura Pearlman. “Use of SAML for OGSI Authorization”, GFD.66. March 2006
20. OASIS. “Security Assertion Markup Language (SAML) 2.0 Specification”, November 2004.
21. Scott Cantor. “Shibboleth Architecture, Protocols and Profiles, Working Draft 10 September 2005, see http://shibboleth.internet2.edu/shibboleth-documents.html
22. OASIS. XACML v3.0 Administrative Policy Version 1.0, Working Draft 16, 22 February 2007
23. N. Zhang, L. Yao, A. Nenadic, J. Chin, C. Goble, A. Rector, D. Chadwick, S. Otenko and Q. Shi; "Achieving Fine-grained Access Control in Virtual Organisations", to appear in Concurrency and Computation: Practice and Experience, published by John Wiley and Sons publisher.
24. For the VPMan project see http://sec.cs.kent.ac.uk/vpman
25. ITU-T Rec X.812 (1995) | ISO/IEC 10181-3:1996 “Security Frameworks for open systems: Access control framework”
26. For the Shintau project, see http://sec.cs.kent.ac.uk/shintau and
27. David W Chadwick, Sean Anthony. “Using WebDAV for Improved Certificate Revocation and Publication”. In LCNS 4582, “Public Key Infrastructure. Proc of 4th European PKI Workshop, June, 2007, Palma de Mallorca, Spain. pp 265-279
28. David W Chadwick, Wensheng Xu, Sassa Otenko, Romain Laborde and Bassem Nasser. “Multi-Session Separation of Duties (MSoD) for RBAC”. First International Workshop on Security Technologies for Next Generation Collaborative Business Applications (SECOBAP'07), April 16-20, 2007, Istanbul, Turkey
29. David W Chadwick, Linying Su, Romain Laborde. “Coordinating Access Control in Grid Services”. Accepted Oct 2007 for publication in Concurrency and Computation: Practice and Experience, John Wiley and Sons.