Persistent versus Dynamic Role Membership

Jean Bacon, Ken Moody

University of Cambridge Computer Laboratory
JJ Thomson Avenue, Cambridge CB3 0FD, UK
email : {Firstname.Lastname}@cl.cam.ac.uk

David Chadwick, Oleksandr Otenko

IS Institute, University of Salford, Salford, M5 4WT, UK
email : {d.w.chadwick@salford.ac.uk, o.otenko@pgt.salford.ac.uk}

Abstract

Various role-based access control (RBAC) models have evolved along with a small number of implementations. One approach is to make role assignment persistent. After authentication, a principal can exercise the privileges associated with the roles it is assigned to without further action. Another approach is that principals must activate roles explicitly when they need to exercise the privileges associated with the roles. In this paper we outline our models and implementations (PERMIS and OASIS), which provide an example of each style of system. Both are concerned with realistic implementations in large-scale, widely distributed systems. We discuss the advantages and disadvantages of persistent versus dynamic role membership.

1 Introduction

Access control is a crucial aspect of most computerised systems. Mandatory schemes have been established for military contexts, and discretionary schemes are associated with the design of most operating systems. When distributed systems were first designed, based on emerging local area network technology, encryption-protected, capability-based access control was often used. Over the years this approach has grown in popularity as systems have become larger and more widely distributed. The capabilities have become encryption protected authorisation certificates managed by service providers, often alongside authentication certificates and integrated with a public key infrastructure.

Managing the access rights of principals to objects becomes increasingly difficult as individual systems grow in size and different systems need to interoperate. The privileges associated with a role are largely independent of the principals who may currently fill that role, and these privileges change slowly as an organization evolves. This is the key idea behind role-based access control (RBAC), in which privileges are assigned to roles rather than to individual principals; that is, the service developer specifies access control rights in terms only of roles.

There are usually many fewer roles than principals in an organisation, although a large organisation may have several thousand roles. RBAC therefore promises to be an appropriate access control scheme for large-scale systems. When RBAC is used, administrators register principals in the usual way, as students, employees etc. It remains to express and enforce the policy that entitles principals to roles.

An additional requirement for managing access control in widely distributed applications is that heterogeneous, independently developed and administered systems should interwork; that is, principals managed by one system will need to use the services of others. Access to such privileges must be negotiated between the systems. Examples are services associated with e-government, where police, social services or health trusts may be authorised to access certain electronic records managed by another agency. Within the health application, the electronic health records of an individual patient may be shared between management domains such as primary care practices, hospitals, clinics etc.

Various RBAC models have evolved over the years [6, 10, 11, 12, 14, 17] but there are few architectures and implementations. If RBAC is to be adopted in practice, large-scale engineering issues must be addressed, one of which is the persistence of the binding of principals to roles. Most of the RBAC models assign principals to organisational roles for an indefinite period. The alternative is that a principal activates a role only when it needs to carry out some task for which the privileges associated with the role are needed. In Section 2 we evaluate the approaches in principle then describe two implemented RBAC schemes, PERMIS in Section 3 and OASIS in Section 4. PERMIS uses persistent role-assignment, OASIS uses dynamic role activation, which we shall call P-RBAC and D-RBAC for brevity. In Section 5 we revisit the two approaches, highlighting the differences that remain significant after design decisions have been taken. We conclude in Section 6.

2 Session-based versus persistent role membership

We aim to evaluate persistent versus dynamic role activation in the abstract and in practice, to the extent that our early implementations allow. Figure 1 gives an abstract architecture for a single service (a) when role membership is persistent (P-RBAC) and (b) when roles must be activated explicitly at the time the service is to be used (D-RBAC). Let us assume that P must be a member of role R in order to have the privilege of invoking service S. Authentication requirements are common to both approaches; let us assume it is carried out by some standard mechanism such as challenge-response.

In (a) principal P's right to role R must be established before service S can be invoked by P. Checking of role membership may be carried out as part of access control; either by checking an administrative database that associates roles with principals or by checking a directory in which some form of signed role membership certificate can be associated with P; both are so-called "pull" approaches and require that P is authenticated. Alternatively, P may keep its role certificate for R in its private file space and may present it to S as a parameter on invocation, a "push" approach. If flexibility is required over whether pull or push will be used in a given implementation then some form of encryption-protected role certificates must be used. Note that environmental checks may also be made as part of access control;

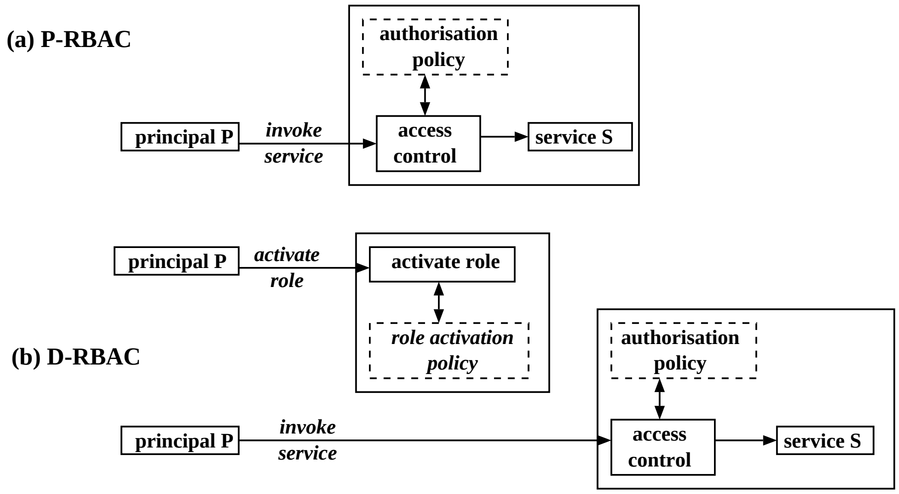

Figure 1: A service architecture with (a) persistent and (b) dynamic role activation

these include time of day and facts that might be looked up.

In (b) P must first activate role R by establishing its right to R according to R's activation policy. The role activation service may look up (pull) P's credentials or P may present (push) them. In both cases the service will authenticate P, to prove its ownership of them. If the credentials satisfy the role activation policy, the service will record that P is active in role R and may also issue a short-lived role certificate to P. When P invokes S, either P presents the role certificate as a parameter and/or the service checks with the role activation service that P has activated role R. Again, environmental checks can be made as required by role activation and service invocation policy.

2.1 Criteria for comparison

Some criteria for comparison and issues that must be addressed by the two approaches are listed below. Later we show how the design decisions taken in PERMIS and OASIS address these issues. For simplicity of discussion, we assume that role certificates are cryptographically signed or protected by encrypted communication. An alternative is to query secure administrative databases for the information.

Role allocation and activation overhead
In P-RBAC principals persistently hold their maximum number of roles; including those inherited by means of RBAC delegation hierarchies. Certificates are issued at a coarse time-grain, e.g. annually, to all principals entitled to them and may be used on many occasions. Many role certificates may never be used.
In D-RBAC roles are activated at need according to role-activation policy. It is likely that only a small proportion of principals and roles will be active at a given time.

The overhead of certificate issuing is therefore a choice between producing large quantities of long-lived certificates, which can be done in batch, offline or background mode (P-RBAC), or producing small quantities of short-lived certificates in the context of a working session (D-RBAC). The tradeoff is application-dependent and the certificates may not be equivalent. D-RBAC certificates may be passed, unsigned, under encrypted communication, for example.

Credentials

In P-RBAC, a principal's right to a role may be proved once-for-all when the role is first issued. Even paper-based credentials are feasible since the role certificates are issued in background mode. In D-RBAC, entitlement of a principal to a role must be proved by electronic means: certificates, smartcards or database records.

Privacy

Some methods of persistently recording principals' assignments to roles might be considered an invasion of privacy for some applications.

Authentication

Both dynamic and persistent RBAC are authentication agnostic and specific implementations may adopt whatever means are appropriate. Note that some systems (Akenti, CAS) mandate the use of PKIs for authentication so that roles can be linked to the principal's public key rather than to the principal's identity. But this is not a feature of P-RBAC or D-RBAC.

Policy expression, enforcement and evolution

P-RBAC and D-RBAC are equivalent with respect to authorisation policy. The policy that entitles principals to roles must also be expressed and enforced in both systems. In P-RBAC, this policy is applied statically, perhaps by a human administrator, and certificates are issued. In D-RBAC, the policy is enforced on role activation.

Policy may require that certain checks must be satisfied, such as time of day, coursework deadlines, relationships between parameters e.g. doctor-patient, and individual exclusions e.g. "all doctors except Fred Smith may read my health record". In P-RBAC assignment of principals to roles is static but environmental checks can be made on service invocation. In D-RBAC environmental checks can be made both on role activation and on service invocation.

Policy will evolve over time, dictated by local and external requirements, and it seems that these changes can be effected more immediately by D-RBAC. There is a need to manage policy change: to control access to policy itself and to ensure consistency as it evolves.

Vulnerability

At first sight it seems that P-RBAC may be more vulnerable to cryptographic attack than D-RBAC. However, the general issue of security of credentials applies to both schemes, involving properties such as the algorithms used and the length of encryption keys. The time available to an attacker should not be an issue if the encryption mechanism is designed well. Vulnerability is more likely to result from theft of keys and certificates. Both approaches are likely to use some long-lived credential, such as a public-private key-pair, for authentication at the start of a session. Vulnerability can be minimised by using session keys instead of persistent keys wherever appropriate, for example to identify a principal within a session.

Revocation

This is a crucial issue for P-RBAC and its importance for D-RBAC depends on how long roles are active. Because few, known principals and roles are active at any time in the latter, revocation seems to be more manageable for it.

Distributed domains
The single-service architecture shown above will in practice require extension to a system of many domains, each comprising many services; for example a national EHR service comprises many primary care practices, hospitals, clinics etc., each with many internal services. It should be possible for a service in a domain to indicate, in its authorisation policy, a role defined in another domain, according to some negotiated agreement.

3 PERMIS overview

PERMIS is a role-based access control infrastructure that is based on standards and driven by policy. Roles are allocated to principals by authorised managers and are held in X.509 attribute certificates. PERMIS has implemented hierarchical RBAC [18], so that a principal who is allocated a superior role will inherit the privileges of all subordinate roles in the role hierarchy. PERMIS has taken a very liberal view of what may be classified as a role. A role is defined by a value of any type (i.e. an attribute in X.509 terminology). So a classical role such as manager, supervisor or employee may be assigned to a principal and, in addition, an application-specific status such as Frequent Flyer Gold, Silver or Bronze. PERMIS provides a Privilege Allocator tool that allows its user to allocate attributes to principals, and to sign the attribute certificates generated with his private key.

Each attribute certificate contains the X.500 distinguished name of the principal (or a pointer to its public key certificate), the attributes (roles) allocated to the principal, the validity period of the certificate, and various other parameters that may limit the applicability of the role assignment. The top level allocating authority is known as the source of authority (SOA) for the embedded attributes. The X.509 standard [19] also specifies mechanisms to control the delegation of authority from the SOA to subordinate attribute authorities, but delegation of authority is not fully supported by the current PERMIS implementation.

3.1 PERMIS Architecture

The integrity and authenticity of attribute certificates are protected by their digital signatures, so they can be stored anywhere without fear of their being modified without detection. Privacy protection is not however provided, so roles of a sensitive nature should not be stored in public repositories. PERMIS has chosen to store the attribute certificates in LDAP directories by default, since LDAP is a popular Internet standard for accessing repositories, and the X.500 standards provide for this. The Privilege Allocator tool can be configured with the URL of an LDAP server, and it will write the attribute certificates directly into the principals' LDAP entries.

PERMIS has been built to support the "pull"or "server pull" model of operation, in which the access control engine pulls a principal's attribute certificates from the LDAP directories whenever it needs to make an access control decision. PERMIS can also support the "push" or "user push model" of operation, in which the principal (or his agent) pushes the attribute certificates to the access control engine whenever a decision is needed (Note that the cur-

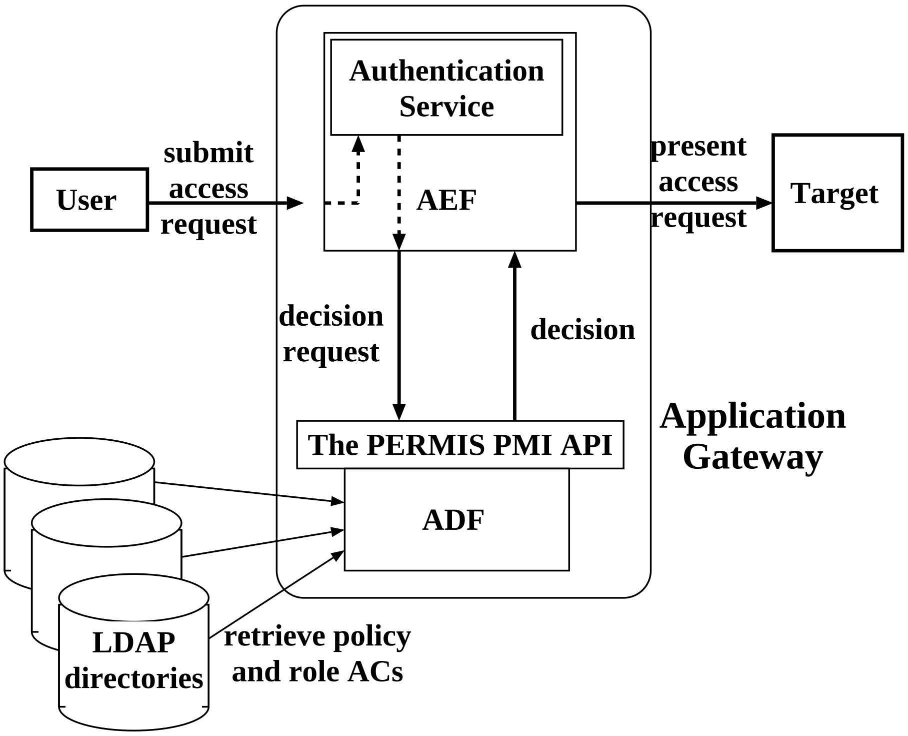

Figure 2: The PERMIS Access Control Architecture

rent release of PERMIS has not implemented the push model fully since it does not check certificate revocation lists, but we plan to add this in a subsequent release.)

The access control engine has been built according to the ISO Access Control Framework (ISO 10181-3) [13]. This splits the access control functionality for an application into two components: the Access Control Enforcement Function (AEF), which is application-specific, and the Access Control Decision Function (ADF), which is application-independent. This architecture ensures that access control decisions within a domain can be consistently enforced by the ADF independently of the application, so that all ADF decisions are based on the access control policy specified for the domain. To support this, PERMIS has defined its own policy grammar (see next section). Each policy of the domain is secured in a policy attribute certificate, digitally signed by the SOA for the domain, and stored in the SOA's LDAP entry (each policy is given a unique object identifier according to the ISO ASN.1 standard). The ADF has been built using Java, and an API has been defined to link the AEF to the ADF. This API comprises three calls (GetCreds, Decision and Finalise) and a constructor. When the ADF is constructed, it reads in the policy for the domain, checks that it has been signed by the SOA for the domain, and that it is the correct policy. Because the AEF and ADF are designed to run as one process, linked via the Java API, there is no need for them to authenticate each other. If they were to run on separate systems with a standalone ADF contacted by multiple AEFs, as in Akenti, then mutual authentication would be necessary. The alternative PERMIS approach is that the ADF runs together with the AEF on each system, and all the ADFs read in the same policy for the domain.

When a user wishes to access an application controlled by PERMIS, he or she must first be authenticated by the application specific AEF. PERMIS has been built to be authentication agnostic. For example, when authenticating a user the AEF could use digital signatures, Kerberos, or even username/password pairs. This is an application specific choice. The only requirement that PERMIS makes is that the principal's distinguished name is passed to the ADF via a call to GetCreds, thus starting a new session. In pull mode, this causes the ADF to retrieve all of the user's attribute certificates from the set of configured LDAP servers. In push mode, the set of ACs is passed at the time of the call to GetCreds. GetCreds validates the user's attribute certificates, discarding any that violate the domain policy, or that have expired, or that have been revoked (in a subsequent release). In addition, any unrecognised or invalid roles/attributes embedded within valid attribute certificates are discarded. When GetCreds has finished, the ADF is left with a set of valid roles that conform to the policy of the domain, and these roles will be used for the duration of the user's session.

Whenever an access request is made by the user, the AEF asks the ADF to make a *granted* or *denied* decision, based on the policy, the target name and the access parameters. In addition to the static policy elements described in Section 3.2, PERMIS includes a plug-in feature so that application specific condition-evaluation objects can be invoked during decision making. Each request that is granted will be passed to the target, whilst each request that is denied will be rejected.

A specific policy is bound to the ADF when it is first constructed. If it is necessary to introduce an alternative policy while an application is executing, the application can destroy the current PERMIS ADF object and construct a replacement. PERMIS also supports an application-defined event object that is called back from the ADF. This allows the AEF to prematurely terminate a user session, for example if the session has been open too long.

3.2 The PERMIS Policy

The PERMIS RBAC Policy comprises the following components:

SubjectPolicy - this specifies the subject domains covered by this policy. Each domain is specified as an LDAP subtree. Only principals with an LDAP distinguished name that falls within a subject domain may be authorised to access targets covered by this policy.
RoleHierarchyPolicy - this specifies the different roles supported by this policy, and their hierarchical relationships to each other.
SOAPolicy - this specifies which SOAs are trusted to allocate roles. The first SOA in the list is the SOA for this domain (and is the one which signs this policy). Subsequent SOAs in the list are remote SOAs which are trusted by the local SOA. This delegates authority to remote SOAs, so supporting distributed role management.
RoleAssignmentPolicy - this specifies which SOAs may allocate which roles to which subjects, whether further delegation of roles may take place, and for how long the roles may be assigned. This sub-policy effectively states who is trusted to allocate which roles to whom, and is central to the distributed management of trust.
TargetPolicy - this specifies the target domains covered by this policy. Each domain is specified as an LDAP subtree. Only domains with an LDAP distinguished name that falls

within a target domain are covered by this policy.
ActionPolicy - this specifies the actions (or methods) supported by the various target domains, along with the parameters that should be passed along with each action, e.g. action *Open* with parameter *Filename*.
TargetAccessPolicy - this specifies which roles may perform which actions on which targets, and under what conditions. Conditions are specified using Boolean logic and may contain constraints such as "IF time is GT 9am AND time is LT 5pm OR IF Calling IP address is of the form 125.67.x.x". All actions that are not specified in a Target Access Policy are denied.

A full description of the policy can be found in [7].

4 OASIS overview

OASIS is an access control system for open, interworking services in a distributed environment, with services being grouped into domains for the purpose of management. Services may be developed independently but service level agreements allow their secure interoperation. OASIS is closely integrated with an active, event-based middleware infrastructure. In this way we can notify applications of any change in their environment, making it possible to ensure that security policy is satisfied at all times. OASIS is role based but has important differences from other RBAC schemes [5, 6, 8, 10, 11, 12, 15, 17, 18]:

- Roles are service-specific; there is no notion of globally centralised administration of role naming and privilege management.
- Roles may be parametrised, as required by applications.
- Roles are activated within sessions. An OASIS session is started by strong authentication of a principal, and an initial role such as *logged_in_user* is created as a side effect of authentication. Roles have activation conditions that may include the requirement for prerequisite roles; a dependency tree of active roles is built up within a session.
- All privileges are associated with roles. We use appointment instead of delegating roles or privileges; the activation conditions of roles may include appointment certificates. Persistent credentials (as opposed to session-limited role membership certificates (RMCs)) are implemented as appointment certificates, which do not confer privileges directly.
- Role activation conditions can also include constraints on the context, which must be checked during role activation.
- We provide an *active security environment*. Certain role activation conditions are specified as having to remain satisfied for the role to remain active. These are monitored and a role is deactivated if any becomes false.

An overview of OASIS is given in [1, 2], details of its architecture and engineering can be found in [3] and a formal model is presented in [4].

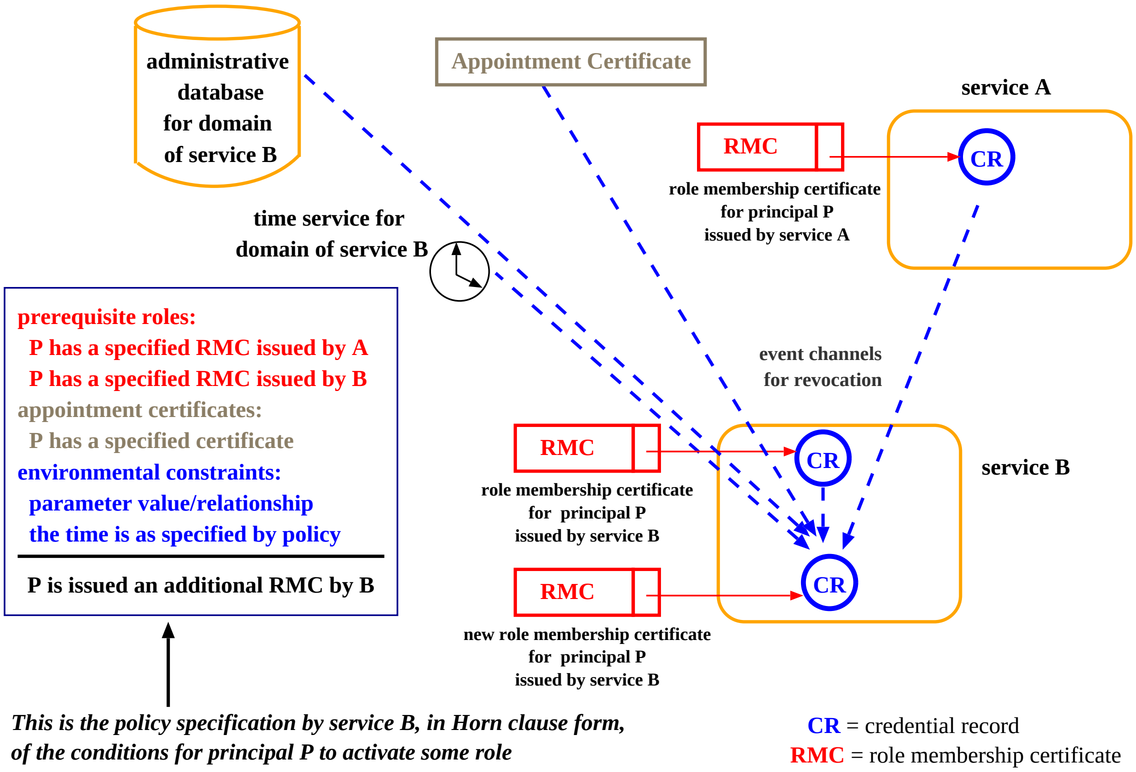

Figure 3: OASIS role activation within a session

In this paper we focus on the implications of dynamic role activation. Figure 3 gives an example of a role activation rule. In order to activate a certain role of service B the principal must have activated two specified prerequisite roles (another role of service B and a role of service A); it must present a persistent credential (an appointment certificate) as proof of some long-lasting employment or qualification; and two environmental constraints must be satisfied. One concerns the time at which the role is activated, the other requires the administrative database to be consulted to determine, for example, some value(s) of or relationship between the parameters of the role (not shown). If these conditions are satisfied the new role is activated, and a role membership certificate (RMC) returned to the principal. A new credential record (CR) is set up by service B to record the status of the role that has been activated.

Certain role activation conditions are flagged as **membership conditions**, that is, they must remain true for the principal to remain active in the role. An event channel is set up associated with each of the membership conditions. Should any become false, this is notified immediately to the service that has issued the RMC.

Figure 4a) is comparable with Figure 1. It shows the architecture of OASIS in principle, in terms of a single service. A service names its clients in terms of roles, issues an RMC on role activation, records the fact in the form of a credential record, and validates RMCs on service invocation. A principal may present RMCs issued by one service in order to satisfy the authorisation policy of another.

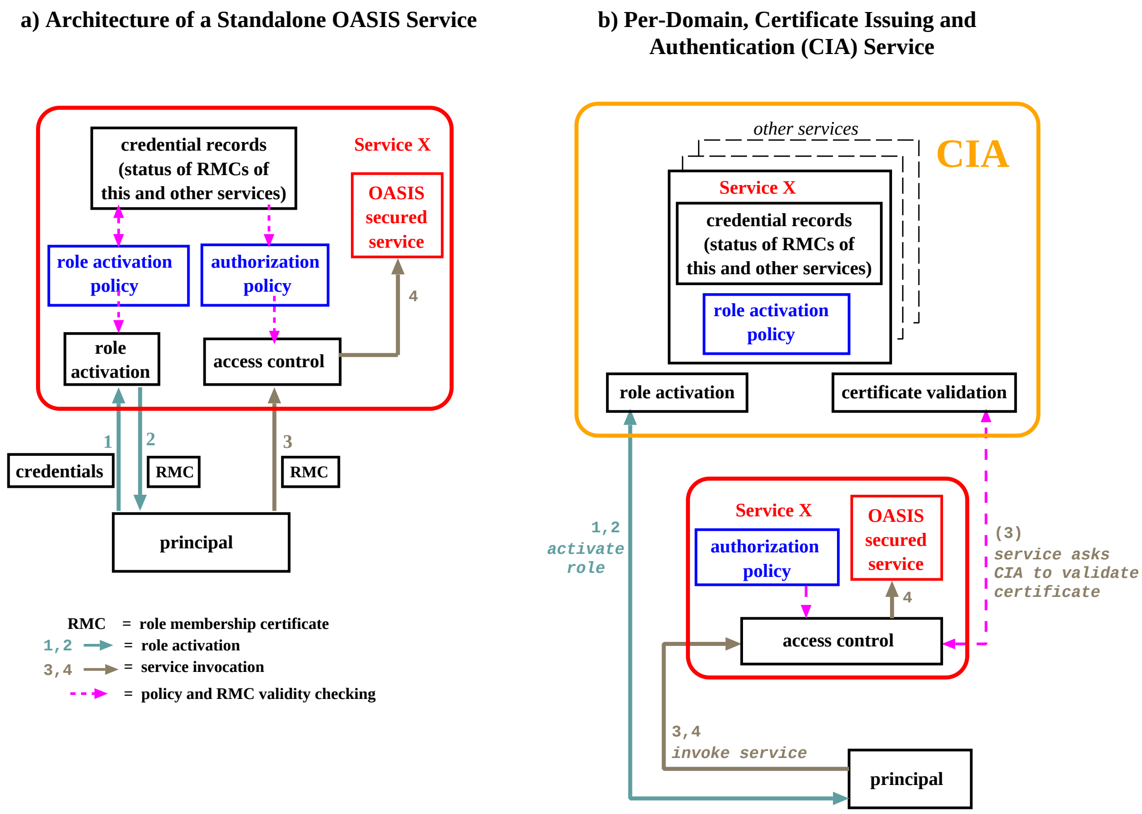

Figure 4: OASIS service architecture and per-domain engineering

In practice, not every service will be capable of this functionality. When OASIS is deployed in large-scale applications we expect each administrative domain to have a secure and available service that takes on the responsibility of role activation, certificate issuing and authentication (CIA) on behalf of all the services in the domain, as shown in Figure 4b). The CIA service is also responsible for monitoring the membership conditions of all the RMCs that it has issued and for maintaining the associated credential records. In our current implementation a web portal includes CIA functionality.

Since the CIA service holds all the active prerequisite links for the services in its domain, membership rule monitoring is greatly simplified. No event channels are needed between services within the domain. Event channels are needed for any membership conditions that are maintained outside the CIA service, such as environmental constraints and any RMCs issued in other domains. In general, the CIA services take over responsibility for most event notification and reception.

Service level agreements can be made so that principals may invoke services in remote domains, while working in their home domain, or may arrange to work temporarily in remote domains and have the right to activate roles there [3]. For example, a hospital doctor may use a national EHR service or may arrange to work temporarily in a Research Clinic.

OASIS RMCs are principal-specific and protected by encrypted communication. In our current, web-based implementation, sessions are managed at a web portal under SSL (secure socket layer). X.509 certificates are used for appointments.

4.1 Appointment

Instead of delegating roles or privileges, and to capture long-lived certification of academic qualification or employment, OASIS uses appointment. A function of some roles is to issue appointment certificates to other principals. Appointment certificates in themselves convey no privileges but may be required by policy as credentials, alongside environmental constraints and prerequisite roles, to activate certain roles.

For example, a professional society or academic institution may issue appointment certificates to people who become professionally or academically qualified. At present a paper certificate is typical, backed up by a database record at the issuing body. Such certificates may be required credentials for activating certain roles; for example, to be a member of an appeals panel at the society or institution.

These certificates may be conditions of employment, such as when a doctor is employed at a hospital. Instead of requiring academic and professional certificates in the activation rules of roles in the hospital, it is preferable for the hospital to issue an appointment certificate indicating "employed as doctor" after the professional and academic qualifications have been validated by the issuing bodies. The internal roles of the hospital need then only require the local "employed as doctor" certificate among their activation rules. This long-lived style of credential may be seen as augmenting a database record. The certificates may be issued as machine-readable cards or may be stored in private or public directories.

Appointment may also be used in situations where a technically-qualified stand-in is required; perhaps someone is called away in an emergency and must arrange a substitute. Here, the appointment certificate is unlikely to be long-lived, but its lifetime is not restricted to the session in which it was issued. The receiver may use the appointment certificate to activate appropriate roles within its own sessions.

5 Persistent versus dynamic role membership

We now revisit the issues introduced in Section 2 in the light of the design decisions taken in PERMIS and OASIS. The implemented systems are closer in functionality than we expected; for example, PERMIS requires roles to be used within sessions and checks role activation policy at the start of a session. A difference in our thinking that has emerged as a result of this study is the assumption on what a role is in our two systems. In PERMIS roles are generic categories, such as doctor, whereas OASIS deems these to be appointments, that is, credentials for activating service-specific, parametrised roles when the need arises. Examples of roles in OASIS are *doctor-on-duty(Dr-ID)*, an initial role requiring strong authentication and an *employed-as-doctor(Dr-ID)* appointment credential, and *treating-doctor(Dr-ID, patient-ID)*, a role required because only a doctor with whom a patient is registered is permitted to read that patient's health record. Role activation policy is likely to require a *doctor-on-duty* prerequisite role and parameter lookup in an administrative database to check the doctor-patient relationship. OASIS was designed to capture fine-grained, service-specific policies of this kind.

Role allocation and activation overhead
Neither implemented system has yet dealt with large numbers of principals and roles e.g. millions of principals and thousands of roles.
For the applications to date, the overhead in PERMIS (P-RBAC), of issuing X.509 role certificates to all principals for all the roles to which they are entitled has been manageable. A Privilege Allocator API allows attribute certificates for all principals to be created automatically from a back-end administrative database e.g. to create "doctor" roles for everyone registered at the General Medical Council. It would not be feasible to create statically the parametrised, service-specific roles envisaged by OASIS, for example a *doctor(Dr-ID, patient-ID)* role for every patient of every doctor. Allocation and management of OASIS persistent credentials (long-term appointment certificates) involves an overhead comparable with allocation of role certificates in P-RBAC, although database lookup can replace the use of appointments in any implementation. Both systems allow certificate allocation to be partitioned between different sources such as professional bodies and academic institutions. OASIS envisages that some generic employment categories, such as doctor or consultant, will be defined in local domains and appointment certificates will be issued on employment, thus subsuming externally issued credentials, some of which may continue to be paper-based. Short-term but session-independent appointment (to enable the delegation of a role) is likely to be used on a small scale. OASIS role activation is on the run-time-critical path and has to be automated. PERMIS role allocation is a background activity.

Privacy
In the PERMIS pull mode persistent role certificates are stored in public LDAP directories, which potentially compromises the user's privacy. If privacy concerns are paramount, then P-RBAC should use the push model and allow the user or Attribute Authority to hold and control the release of his certificates across a confidential channel. The downside of allowing the user (rather than the AA) to control his role certificates is that revocation checking must now be carried out by the ADF. In D-RBAC the role certificates are not publicly visible when issued or presented and pass within communications that are encrypted under SSL. Thus privacy is ensured.

Vulnerability
Both projects lack experience over a long term of deployment. Both are using standard PKI services: SSL (OASIS), X.509 certificates for roles (PERMIS) and appointments (OASIS).

Revocation
D-RBAC does not necessarily need a revocation mechanism because the roles, and therefore the privileges, are short-lived. But since only a relatively small number of principals are active at any time, it is practicable to have role membership rules (that indicate which role activation conditions must remain true for the principal to remain active in the role) and to monitor that they remain satisfied. Immediate revocation of role certificates is part of the OASIS scheme, effected by invalidating the credential record for the RMC, and notifying any service that has cached its validity and requested call-back.

In P-RBAC it is not feasible to monitor every persistent role certificate in a system continuously, but this is not required, since only currently active certificates need to be monitored. In PERMIS initial monitoring is done at session initialisation time, by either retrieving the latest certificate revocation list or issuing an OCSP (online certificate status protocol) [16] request. Continuous monitoring could be achieved by using a protocol such as OCSP, requiring OCSP queries to the OCSP server immediately prior to each access control decision, but this would be inefficient. It should be necessary only for transactions where the loss associated with accepting a transaction, for a role certificate that has been revoked after initial monitoring, is high. Instead, PERMIS has a callback procedure that allows the AEF to terminate the user's session prematurely in an application-specific way.

Ease of use
The PERMIS P-RBAC pull model is easy for the principal to use, since he does not need to know about role certificates, how many he has, or where they are stored. The service will take care of them by pulling them from the public repositories. The principal merely needs to know how to authenticate to a particular service. The PERMIS push model requires the principal or his AA to know about and to actively manage his role certificates. This may or may not be a problem for the average user, but as we currently do not have any empirical data of user experience, it is too early to say.

The current OASIS implementation, for a web service environment, requires OASIS services and user agents to have PKI facilities. The web portal acts as session manager. For a given authorisation policy it acquires the necessary roles on behalf of the user (validating each role's activation policy) and presents role certificates under SSL.

In both systems, policy is specified by the service developer and may include environmental checks with application and/or domain databases. As noted above, PERMIS and OASIS roles differ and there may be a tradeoff between simplicity of policy expression and expressiveness/flexibility.

Distributed domains
The standards underlying the PERMIS system for P-RBAC have been designed for distributed domains and the federated management of role certificates. Standard inter-domain communications protocols have been used for the transfer of role certificates and revocation information, e.g. LDAP, OCSP, SAML.

In OASIS D-RBAC, there is additional overhead when role certificates need to be validated at an external domain, by callback to the issuer, and provision must also be made for revocation. In OASIS this is handled by integrating the access control system with an asynchronous platform. The overheads include setting up an event channel and establishing a heartbeat protocol, usually between the respective CIA servers.

6 Conclusions

In comparing our implementations in detail we found many similarities. Even though PERMIS certificates are long-lived, and their format is not session-specific, principal-role assignments are checked at session granularity. For ease of use, the most recent implementation of OASIS, for a web-based environment, has moved session and (unsigned) certificate management into a web portal. This is similar to the PERMIS and ISO access control architectures.

At any time only a small proportion of principals are active in roles. Roles that are allocated in P-RBAC may never be used, but this inefficiency may be small compared to that of repeatedly creating the same role certificates for principals in D-RBAC. If the roles defined in P-RBAC and D-RBAC were identical, and the certificates of equivalent weight, this argument would be compelling. However, OASIS certificates are not signed, since thay are protected by SSL encryption, and its parametrised roles were designed to capture complex policies involving, for example, excluded principals and relationships between principals. The thesis that dynamic, service-specific roles are better able to capture and enforce such policy remains to be tested in real applications.

Delegation of authority poses difficulties for P-RBAC, such as what to do with a delegatee's role when the delegator loses the delagating role through change of function or dismissal. The traditional policy in organisations is that the delegatee's role should remain valid, but normal certificate chain validation rules would make it invalid. This issue does not arise in D-RBAC, since role certificates are created dynamically when they are needed.

OASIS was designed to support instant role revocation, incurring some overhead, yet this is arguably less critical for D-RBAC than for P-RBAC, where long-term monitoring is not feasible. PERMIS addresses this problem by carrying out principal-role validity checks when persistent roles are used for service invocation.

A key difference relates to enforcing the principle of "minimum necessary privilege". If principals hold all their role certificates in P-RBAC, and decide which to present within some session, then it is hard to check and enforce security policies such as static and dynamic separation of duties (SoD) system-wide. However, if role certificates are stored in central repositories, then SoD can more easily be enforced. D-RBAC can implement and enforce these policies more easily than either P-RBAC scheme.

We believe that the important design decisions for the future will be concerned with scalability and the expression, enforcement and management of complex, evolving, distributed policy. Acceptance by users, application developers and administrators will be critical. Neither system has yet been tested on a realistic scale, for example associated with nationwide systems comprising millions of principals, thousands of domains and many thousands of roles. We look forward to gaining experience from the use of our implementations in large-scale applications.

Acknowledgements

The UK Engineering and Science Research Council have supported two graduate students working on OASIS and a Research Fellow working on the application of PERMIS to Electronic Prescription Processing. The EC ISIS program has partially funded the PERMIS project.

References

[1] J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, and M. Spiteri. Generic support for distributed applications. *IEEE Computer*, pp.68–76, March 2000.

[2] J. Bacon and K. Moody. Towards open, secure, widely distributed services. Comm ACM, 43(6), pp.59–63, June 2002

[3] J. Bacon, K. Moody and W. Yao. Access Control and Trust in the Use of Widely Distributed Services. In *Middleware 2001*, Lecture Notes in Computer Science 2218, pp.295–310, Springer 2001

[4] J. Bacon, K. Moody, and W. Yao. A model of OASIS role-based access control and its support for active security. ACM Transactions on Information and System Security, Vol. 5, No. 4 (November), pp.492–540, ACM Press, New York, NY, 2002.

[5] E. Barka and R. Sandhu. A role-based delegation model and some extensions. In *23rd National Information Systems Security Conference*, NISSC 2000, Baltimore, MD, Oct 2000.

[6] E. Barka and R. Sandhu. Framework for role-based delegation models. In *16th Annual Computer Security Applications Conference*, ACSAC 2000, New Orleans, LA, pp.168–177, Dec 2000.

[7] D. W. Chadwick and A. Otenko. The PERMIS X.509 role-based privilege management infrastructure *Proceedings, Seventh ACM Symposium on Access Control Models and Technologies (SACMAT)*, Monterey CA, pp.135–140, June 2002.

[8] M. J. Covington, M. J. Moyer, and M. Ahamad. Generalized role-based access control for securing future applications. In *23rd National Information Systems Security Conference*, NISSC 2000, Baltimore, MD, Oct 2000.

[9] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas and T. Ylonen. SPKI Certificate Theory Internet RFC 2693, September 1999

[10] D. F. Ferraiolo, J. F. Barkley, and D. R. Kuhn. A role-based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information and System Security*, 2(1):34–64, Feb. 1999.

[11] L. Giuri and P. Iglio. Role templates for content-based access control. In *Second ACM Workshop on Role-Based Access Control*, pages 153–159, Fairfax, Virginia, November 1997.

[12] C. Goh and A. Baldwin. Towards a more complete model of role. In *Third ACM Workshop on Role-Based Access Control*, pages 55–61, Fairfax, Virginia, October 1998.

[13] ISO/IEC 10181-3:1996 Open Systems Interconnection, Security frameworks for open systems: Access control framework

[14] M. Koch, L. V. Mancini and F. Parisi-Presicce. On the specification and evolution of access control policies. In *Proceedings, Sixth ACM Symposium on Access Control Models and Technologies (SACMAT)*, Chantilly, VA, pp.121–130, May 2001

[15] J. D. Moffett and E. C. Lupu. The uses of role hierarchies in access control. In *Fourth ACM Workshop on Role-Based Access Control*, pages 153–160, Fairfax, Virginia, October 1999.

[16] M. Myers, R. Ankney, A. Malpani, S. Galperin and C. Adams. X.509 Internet PKI Online Certificate Status Protocol - OCSP. IETF RFC 2560, June 1999.

[17] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-Based Access Control Models. *Computer*, 29(2):38–47, Feb. 1996.

[18] R. Sandhu. Role activation hierarchies. In *Third ACM Workshop on Role-Based Access Control*, pages 33–40, Fairfax, Virginia, October 1998.

[19] S. Tuecke et al. Internet X.509 Public key Infrastructure - Proxy Certificate Profile. ¡draft-ietf-pkix-proxy-03¿, April 2003