Access control (authorisation) in distributed systems

recall lecture 9 - Introduction to DS: slides 21 to 27
for access control within the overall system architecture:

- as an individual e.g. from home
- within a single administration domain e.g. CL
- using external services from a domain as an individual or group member
- federated domains: inter-domain authorisation

We are concerned with authorisation for service use and/or object access
How is access control policy expressed and enforced?

Authorisation and authentication

Authorisation is built above authentication (proof of identity – proof that you are who you say you are – will someone/something vouch for you?).

Within an administration domain, principals are named and registered as individuals and members of groups.

Principals authenticate in their home domain by means of e.g. passwords.

The aim is to avoid having to have a ***username/password*** for every service.

(.... *How does one remember them all?* ......

....*Use the same one for all? No, break one break all* .....)

A Single Sign On service is needed.

Authentication is covered in Security courses.

For background reading, slides 29-36 outline some single sign on systems for cross-domain service use: Raven, Shibboleth, OpenID

Access control – from first principles

Model: access matrix A (i, j) rows represent principals, columns objects
entry (i, j) contains the rights principal i has to object j

Implementation: since the matrix is sparse (most entries are null)

1. Using access control lists (ACLs): keep non-null entries of column j with object j
   ACL entry = principal name + access rights
   optimisation: group name = list of principals

2. Using capabilities: keep non-null entries of row i with principal i
   a capability (capability list entry) = object name + access rights

Assume managers for the various types of object

On an access request, the manager must check that the requesting principal has the appropriate right to access the object

1. check that the ACL list contains an entry for that principal with the right
2. check that the capability passed by the principal with the request contains the right

ACLs – cf. - capabilities

ACLs

Expressiveness: subtle expression of policy – entries may be for individual principals and groups with individual exceptions.

Revocation: easy to revoke – but ACL changes may not have immediate effect (because the ACL may not be checked on every access once an object is open)

BUT: slow to check - scalability problem

- if expressiveness exploited e.g. negatives and exceptions allowed
- if there are many principals and large groups
- generalisation? multi-domain operation? names outside domain of registration?

AND: awkward to delegate rights e.g. For a file to a printer for a single print job
In a distributed system many services are not part of privileged OSs.

Capabilities

Quick to check – like a ticket – so scale well

Anonymous – knowledge of names not needed – may generalise to multiple domains.
- anonymity may be wanted by some applications for privacy reasons

Problems/issues ... because they are associated with the process rather than the object ...

Capability-based access control - issues

as defined so far, a capability is an object name and some rights

1. protection

   must prevent unauthorised creation, tampering, theft

2. control of propagation

   can principals pass on copies?
   must they ask the object manager? How can this be enforced?

3. delegation

   is an example of propagation
   often with restricted rights for a limited time or action

4. revocation

   - if the access control policy changes, and certain principals should lose their rights, their capabilities should ideally be revoked. Can this be done without revoking all capabilities for the service/object?
   - if a capability is known to have been stolen or tampered with it should be revoked instantly. Will this invalidate all capabilities for this service/object?
   Anonymity has created revocation problems.

Capabilities in centralised and distributed systems

centralised

Several capability architectures were designed and built e.g. Plessey PP250, CAP
Capabilities can be protected by the hardware and/or the OS
A capability is named via an index into a segment or an OS table.
- held in protected OS space per process
- held in typed capability segments in user space with operations such as
  insert, delete, use-as-argument

distributed

Can't be protected by hardware/OS
Have to be transferred across networks and pass through user space
so must be encryption-protected
Security terminology and implementation:
capability = signed certificate
e.g. X.509 authentication and attribute certificates

Capabilities in distributed systems - design

Must be protected by encryption

The object manager (certificate issuer) keeps a ***SECRET*** (random number, private key) and uses a well-known function ***f***, a one-way function.

A capability is constructed using

check digits = f (SECRET, protected fields)

protected fields	***check digits*** (signature)

When a capability is presented with an operation invocation, the manager checks that:

f (SECRET, protected fields) = check digits

If not, the invocation is rejected.

More generally, the invoked service may not be the capability issuer. The service can check back with the issuer (cf. Certification Authority)

Encryption-protected capabilities – issues?

1. protection

   protect against tampering – adding rights,
   NOT against theft – eavesdropping on the network and replay attacks

2. control of propagation

   still no control over propagation

3. delegation

   object manager must be asked to create a capability with reduced rights to pass to another principal for delegated authority.
   Works indefinitely – duration is not controlled – nor further transfer

4. revocation

   (recall: needed when access control policy changes as well as for stolen capabilities)
   - expiry time as a protected field (like X.509) – crude mechanism
   - hot list of invalid invoking principals per service/object (spoofing? check overhead?)
   - change the SECRET – not selective – all old capabilities will not work and authorised principals will have to request new capabilities.

Principal-specific capabilities

include the name of the principal in the capability for generation and checking as an argument to ***f***, perhaps as a protected field in the capability.

1. protection

   from tampering – YES,   from theft – YES: authenticate presenting principal

2. control of propagation

   YES – a capability for the receiving principal can only be created by the object manager

3. delegation

   YES – a capability for the receiving principal must be created by the object manager

4. revocation

   can be more selective – still involves overhead of checking hot list
   e.g. revocation list of principals excluded by policy change
   stolen capabilities should be detected on authenticating the presenting principal
   unless the presenter is successfully masquerading as the owner.

All the above raise the question of the structure and scope of principal names and how and where principals are authenticated.

ACLs in distributed systems

We first followed the capability thread after slide 11.

We have discussed principal-specific capabilities

Now, return to consider ACLs.

ACLs comprise lists of principals (or groups)

ACL entry = principal (or group) name, rights (from slide 3)

Where principals and groups are defined and registered within some administration domain.

Without group names ACLs may become unmanageable, long lists of principals.

Within the administration domain where a group and its constituent principals are registered, a group name can be expanded into a list of principals, for checking.

How can group names be used outside the domain where the group is registered?

We generalise groups to roles and consider role-based access control (RBAC)

Role-based access control (RBAC)

Services may classify their clients into named roles e.g.

login service: ***logged-in-user*** (after authentication)

patient monitoring service: ***surgeon, doctor, nurse, patient***

online exam service: ***candidate, examiner, chief examiner***

digital library service: ***reader, librarian, administrator***

Access rights (privileges) are assigned to roles for use of services

(method invocation) or more fine-grained access to individual objects or broad categories of object managed by a service

Scope of role names may be the local domain of the service, or some role names may be organisation-wide, across federated domains

e.g. ***sales-manager*** used in all branches of a world-wide company

police-sergeant used in all of the 52 UK county police forces

NHS-doctor used throughout the UK NHS

RBAC - 2

Administration: note the separation:

principals* –> *roles*,   *roles* –> *privileges

Service developers need only specify authorisation in terms of roles, independently of the administration of principals

e.g. annual student cohort, staff leaving and joining

Principals are authenticated, as always, and must also prove their right to acquire/activate a role. They thus prove they are authorised to use a service

Compare with ACLs – like ACLs containing only group names.

Compare with capabilities – can a capability that proves role membership be engineered?

RBAC seems promising for fast authorisation checking.

RBAC – 3: Parametrised roles

Roles may be parametrised for fine-grained access control to capture:

- relationships between principals:

Policy: “*only the doctor treating a patient may access the medical record*”

e.g. ***treating-doctor (hospital-ID, doctor-ID, patient-ID)***

- patients and others may express exclusions as authorisation policy

e.g. ***doctor (doctor-ID)***

Policy: “*where doctor is not Shipman*”, “*where doctor is not <x> (a relative)*”

Compare with ACLs containing only groups, with exclusions of individual members

– semantics of precedence of evaluation in ACLs has always been a difficult area.

RBAC – 4: Role hierarchies

Some RBAC systems define role hierarchies with privilege inheritance up the hierarchy.

The hierarchy may mirror organisational structure, which reflects power and responsibility rather than functional competence.

Privilege inheritance is even less defensible for functional roles.

Also: privilege inheritance violates the *principle of minimum necessary privilege*
and makes reasoning about privileges difficult
– see many ACM SACMAT papers)

Role hierarchies are defined in the later NIST RBAC standards.

Our work has avoided privilege inheritance (see OASIS case study).

RBAC – 5: Inter-domain authorisation

RBAC eases authorisation outside principals' home domains, because:

- Roles change less frequently than principals leave and join them
- Administration of users and role membership is separate from service development and use.
- Negotiation on use of services external to domains can be in terms of roles, e.g. payment for a role to use a service
- Federated domains may contain agreed role names in each domain.
  Makes policy easier to negotiate and express.
  e.g. ***sales-department-staff, sales-manager, salesman***

RBAC – 6: Authorisation context

Authorisation policy could include other constraints on use of a role e.g. time of day, as well as relationships and exclusions.

see OASIS case study – environmental constraints

The privileges associated with a role might not be static.
e.g. ***student (course-ID, student-ID)*** may read solutions to exercises only after marked work has been returned.

e.g. Conference management system – a small-scale example follows of use of an external service from a number of domains.

Example: conference management (e.g. Easychair, CMT, EDAS, ...) selection from workflow and policy

Program chair registers names, email addresses, initial password, and roles of the programme committee: roles ***PC-chair(s), PC-member***

all are sent an email asking them to register their account, change their password

Authors submit papers, acquiring role ***contact-author***, returned a UID for the paper

contact-author may submit new versions up to the deadline

PC-members are assigned papers to review. They may delegate some reviews:

role ***reviewer*** per paper, separate from ***PC-member***

Conflicts of interest must be expressed by submitting-authors and PC members, and enforced by the system

PC members must never be able to know the reviewers and see the reviews of their own papers

PC members can see only their own reviews until after the review deadline.

After this, in a discussion phase, PC members may be able to see the ranked order and other reviews (except for their own papers). Systems vary in this respect.

Note: small scale example e.g. 50 PC members, 200 papers

Note: rights will change after deadlines (an example of context)

Design of capabilities/certificates can incorporate RBAC

Traditional capabilities in centralised systems:

object-ID	***rights***

proves the presenter has the rights to the object

RBAC

role	***parameters***

proves the presenter holds the role + parameters
must be checked against access control policy

Capabilities/certificates in distributed systems

check digits = ***f*** (SECRET, protected fields)

protected fields (***object-ID, rights***)	***check digits*** (signature)

RBAC in distributed systems

check digits = ***f*** (SECRET, protected fields)

protected fields (***role, parameters***)	***check digits*** (signature)

RBAC - discussion

RBAC provides:

1. Expressiveness
   - subtle expression of access control policy.
   - if roles are parametrised, exclusions and relationships can be captured.
   - environmental/context checks (time/place) can also be included.

2. Efficiency
   - checking faster than ACLs
   - use of certificate technology comparable with capabilities
   - or use a secure channel and role authentication in source domain

3 Cross-domain interworking
   - easy to negotiate
   - authorisation policy expressible and enforceable
   - heterogeneity of certificates – can check back with issuing domain

OASIS RBAC
Open Architecture for Securely Interworking Services
Case study from Opera Group research

- OASIS services name their clients in terms of **roles**

- OASIS services specify **policy** in terms of **roles**
  - for **role entry** (activation)
  - for **service invocation** (authorisation, access control)

  both in Horn clause form

see: *www.cl.cam.ac.uk/Research/SRG/opera*
for people, projects, publications for download

OASIS model of **role activation**

a role activation rule is of the form:

condition1, condition2, ….. |- target role

where the conditions can be

- prerequisite role
- appointment credential
- environmental constraint

all are parametrised

OASIS (continued) **membership** rules

as we have seen, a role activation rule:

cond1*, cond2, cond3*, ….. |- target role

role membership rule:

the role activation conditions that must **remain true**, e.g.*

for the principal to remain active in the role

monitored using **event-based middleware**

another contributor to an **active security environment**

OASIS model of **authorisation**

An authorisation rule is of the form:

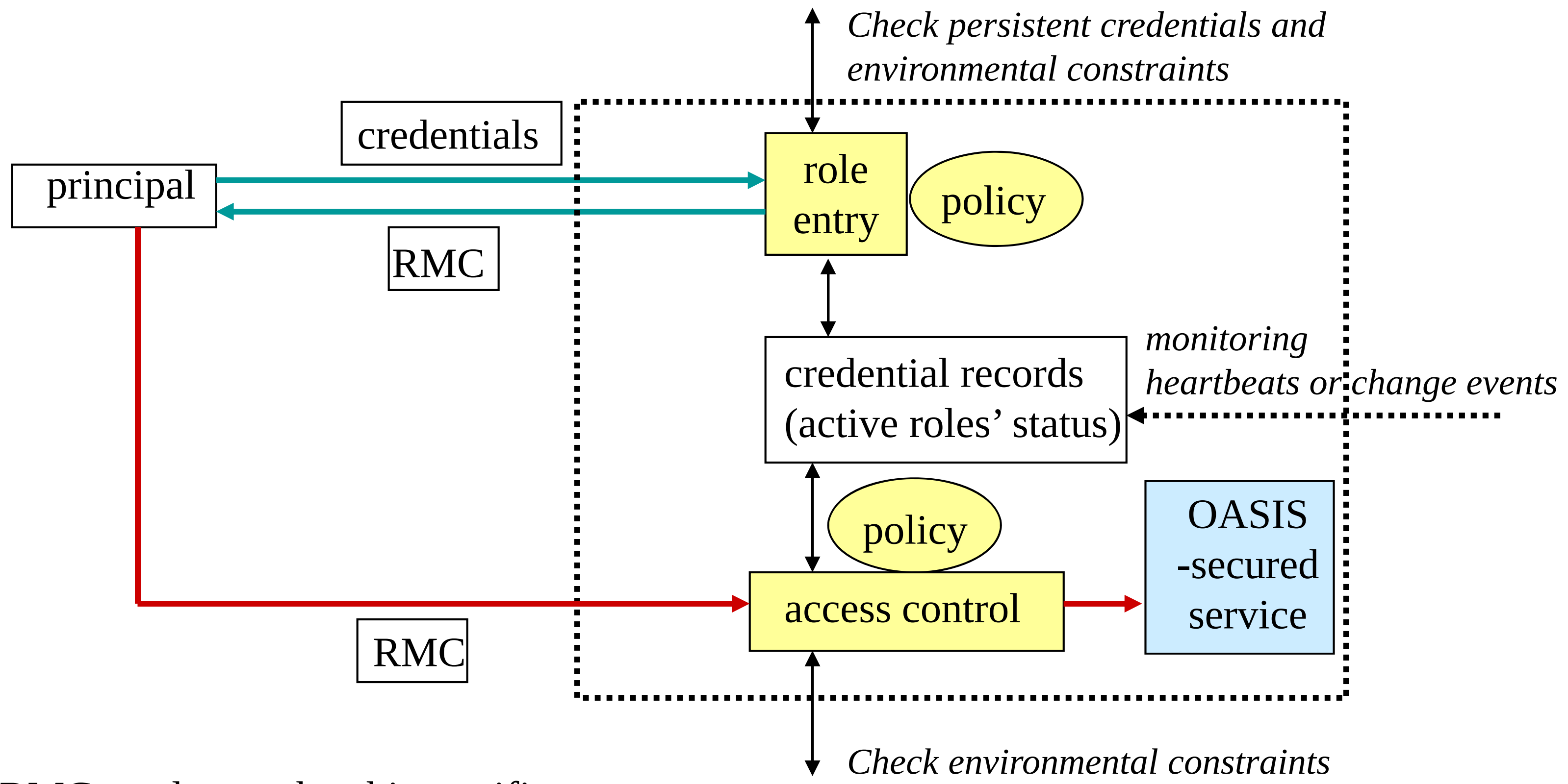
condition1, condition2, ….. |- access

where the conditions can be

- an active role
- an environmental constraint

all are parametrised

A Service Secured by OASIS Access Control

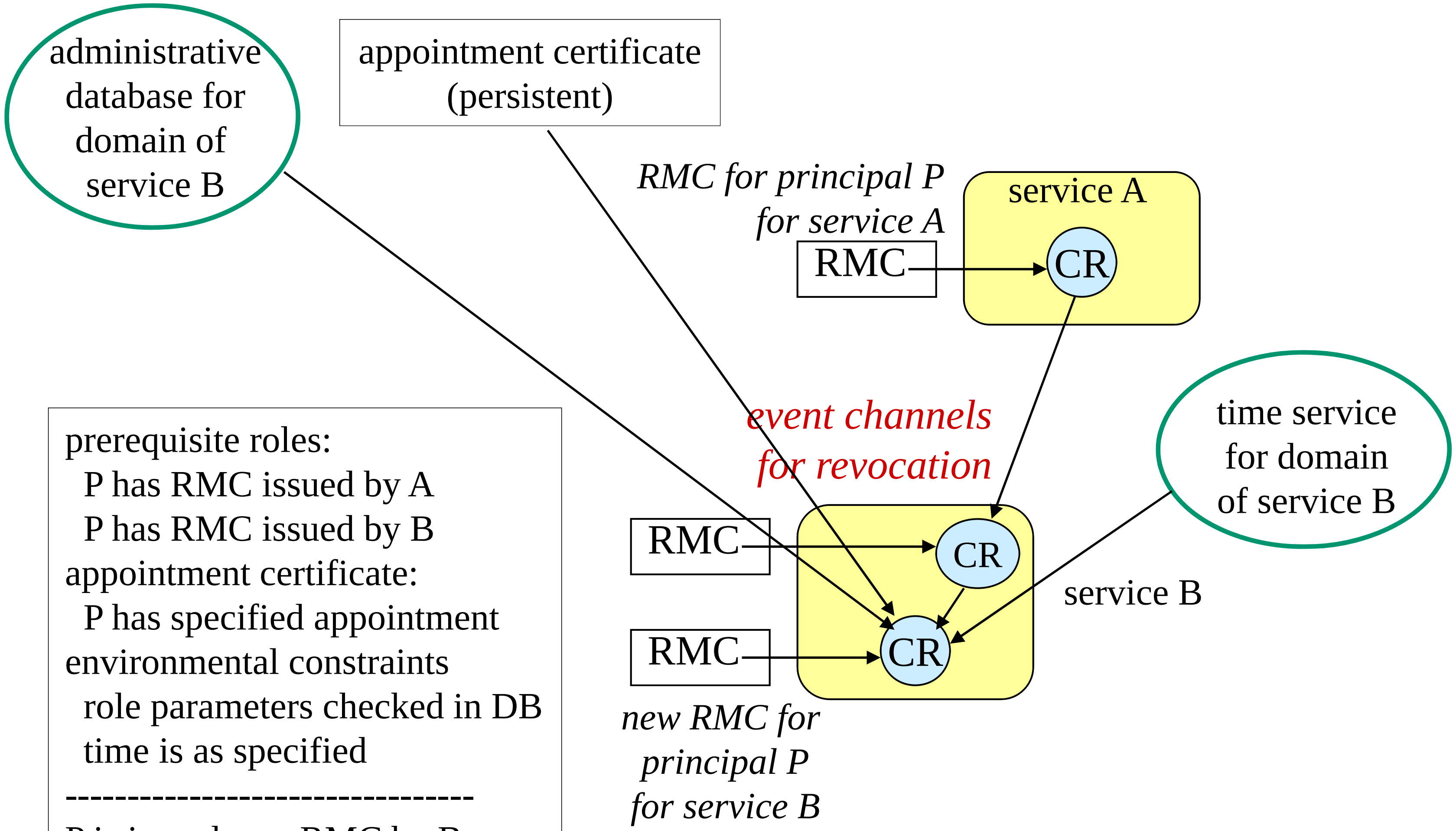

Check persistent credentials and environmental constraints
credentials
principal
role entry
policy
RMC
credential records (active roles' status)
monitoring heartbeats or change events
policy
OASIS -secured service
access control
RMC
Check environmental constraints

RMC = role membership certificate

→ = role entry

→ = use of service

OASIS role activation illustrated

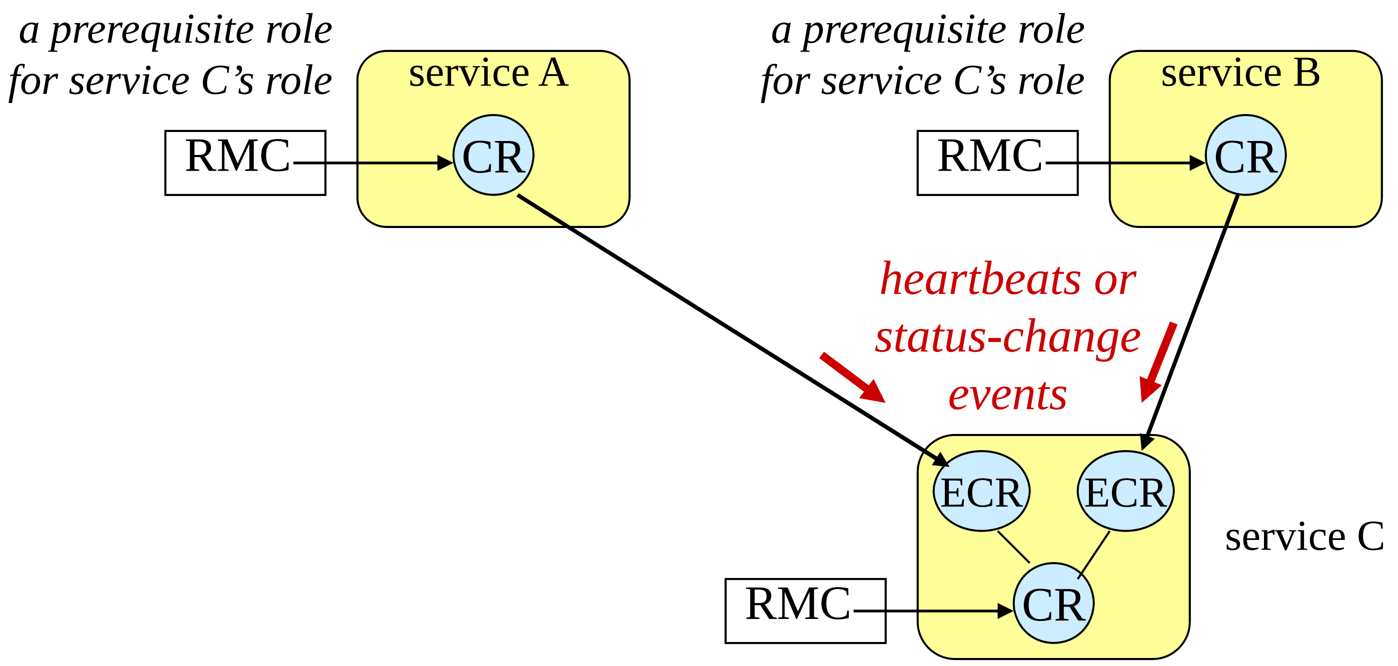

administrative database for domain of service B
appointment certificate (persistent)
RMC for principal P for service A
service A
RMC
CR
event channels for revocation
time service for domain of service B
RMC
CR
RMC
CR
service B
new RMC for principal P for service B

prerequisite roles:
  P has RMC issued by A
  P has RMC issued by B
appointment certificate:
  P has specified appointment
environmental constraints
  role parameters checked in DB
  time is as specified

P is issued new RMC by B

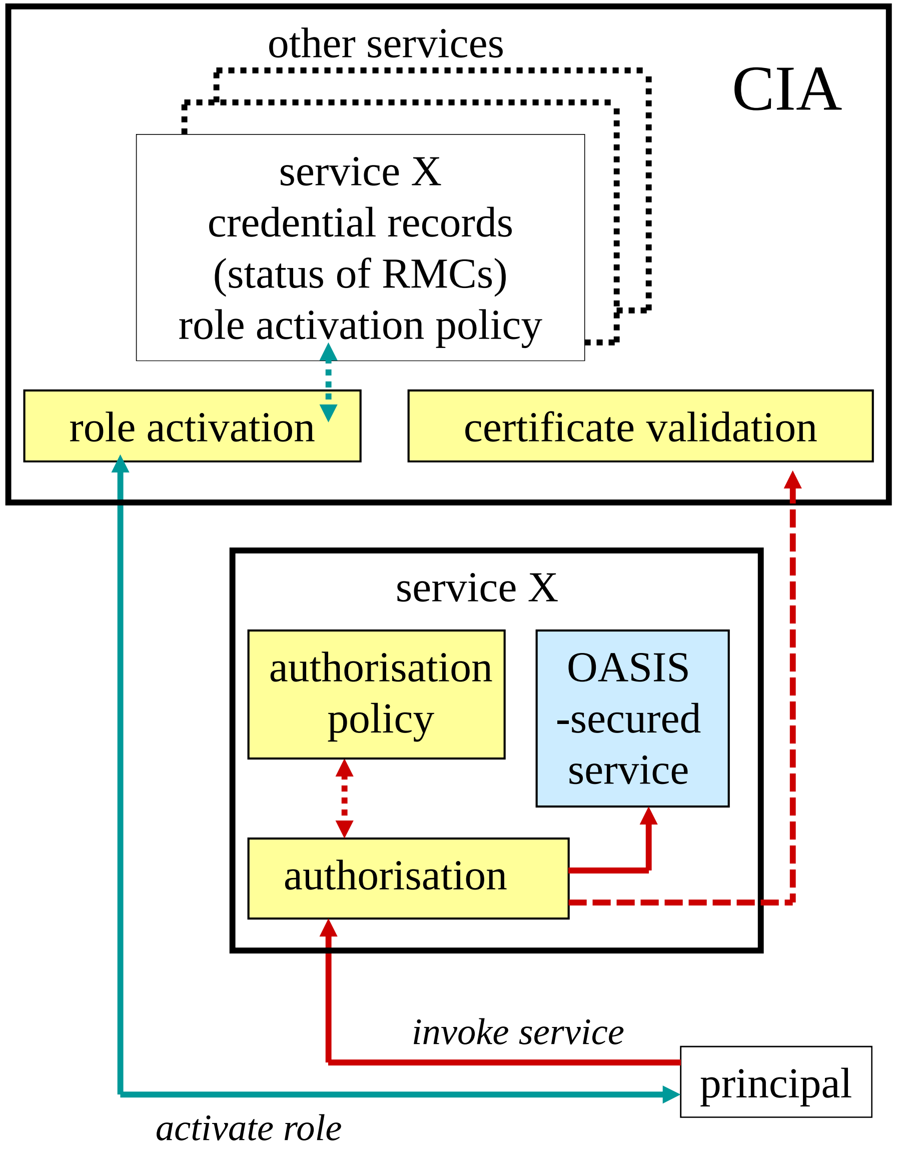
role entry policy specification of service B, in Horn clause form
conditions for principal P to activate some role

Active Security Environment
Monitoring membership rules of active roles



RMC = role membership certificate
CR = credential record
ECR = external credential record

Engineering per-domain certificate issuing and authentication



It is not realistic for every service to manage secrets and issue certificates

The CIA service, for services in its domain:
- *keeps the activation polices*
- *activates roles*
- *issues and validates certificates*
- *maintains credential record structures for active roles*
- *handles revocation via event channels*

The CIA service, for services in other domains:
- *validates certificates it has issued*
- *handles revocation of its certificates*

OASIS philosophy and characteristics

- Distributed architecture, not a single organisation. Incremental deployment of independently developed services in independent administration domains.
- RBAC for scalability, parametrised roles for expressiveness of policy (e.g. exclusion of values, relationships between parameters).
- Policy expression is per service, per domain
- Roles are activated within sessions. Persistent credentials may be required for role activation.
- Independent designs of RMCs may coexist – service at which RMC is presented checks back with issuer for RMC validation
- Service (domain) level agreements on use of others' RMCs
- Anonymity if and when required
- Immediate revocation on an individual basis
- No role hierarchies with inheritance of privileges

Background on cross-domain authentication

(From slide 2) – here is an outline of some single sign on systems

Raven for use of websites across all the domains of Cambridge University
- common naming of principals (CRSIDs, nested domains)
- authentication is sufficient for authorisation

Shibboleth organisation-centric.
organisation negotiates use by its members of external services

OpenID user-centric
used by many large websites (BBC, Google, MySpace, PayPal, ...)

Raven

- Aim: avoid proliferation of passwords for UCam web services
  - Raven is a Ucam-webauth Single Sign On system instance
  - Developed within Cambridge (by Jon Warbrick)

- Three parties in the Ucam_webauth protocol:
  - User’s web-browser
  - Target web-server
  - Raven web-server

- Authentication token passed as an HTTP cookie
  - Thus should be passed using HTTPS… but often isn’t

Example Raven dialogue

- User requests protected page
- Target web-server checks for Ucam-WLS-Session cookie
- If found, and decodes correctly, page is returned. Done.

- Otherwise, redirect client browser to Raven server
  - Encodes information about the requested page in the URL
- Raven inputs and checks credentials
  - (Also permits users to “cancel”)
- Raven redirects client browser to the protected page. Done.
  - (An HTTP 401 error will be generated if users cancelled)

Raven coordinates participants using time

- Target web-server verifies Ucam-WLS-Session cookie
  - Public-key of Raven server pre-loaded on target web-server
- Target web-server and Raven do not interact directly
  - Client browser receives, stores and resends cookies
- What about malicious client behaviour or interception?
  - e.g. replay attacks?
- Raven requires time-synchronisation
  - A site-specific clock-skew margin can be configured

Shibboleth provides federated authentication

- System for federated authentication and authorisation
  - Internet2 middleware group standard
  - Implements SAML: Security Assertion Markup Language
  - Facilitates single-sign-on across administrative domains

- Raven actually speaks both Ucam-webauth and Shibboleth
  - Shibboleth has the advantage of wider software support

- Identity providers (IdPs) supply user information
- Service providers (SPs) consume this information and get access to secure content

Shibboleth exchange

- Similar to Raven, but with some extra indirection
  - User requests protected resource from SP
  - SP crafts authentication request
  - User redirected to IdP or ‘Where Are You From’ service
    - E.g. UK Federation WAYF service
  - User authenticates (external to Shibboleth)
  - Shibboleth generates SAML authentication assertion handle
  - User redirected to SP
  - SP may issue AttributeQuery to IdP’s attribute service
  - SP can make access control decision

OpenID

- Another cross-domain single-sign-on system

- Shibboleth is organisation-centric
  - Organisations must agree to accept other organisations’ statements regarding foreign users
  - Lots of support within the UK Joint Information Systems Committee (JISC) for accessing electronic resources

- OpenID is user-centric
  - Primarily about identity
  - OpenIDs are permanent URI or XRI structures

OpenID (cont)

- User provides their ID to relying party web site
  - OpenID 1.0 retrieves URL, learns identity provider
  - OpenID 2.0 retrieves XRDS, learns identity provider
    - XRDS/Yadis indirection affords greater flexibility

- Many big commercial players offer OpenID assertions
- Lots of open source software support for OpenID also

- In terms of responsibility, consider use for:
  - Access to a web resource
  - Access to a wireless network