Cambridge Event Architecture

synchronous → asynchronous
middleware platform

composition of events

composition of applications using events

storage and retrieval of events

OASIS: Open Architecture for Secure Interworking Services

OASIS: Open Architecture for Secure Interworking Services

* Oasis services name their clients in terms of roles

* Oasis services, and Oasis-aware services, specify policy in terms of roles
  - for service use (method invocation)
  - for object access *cf. ACL*

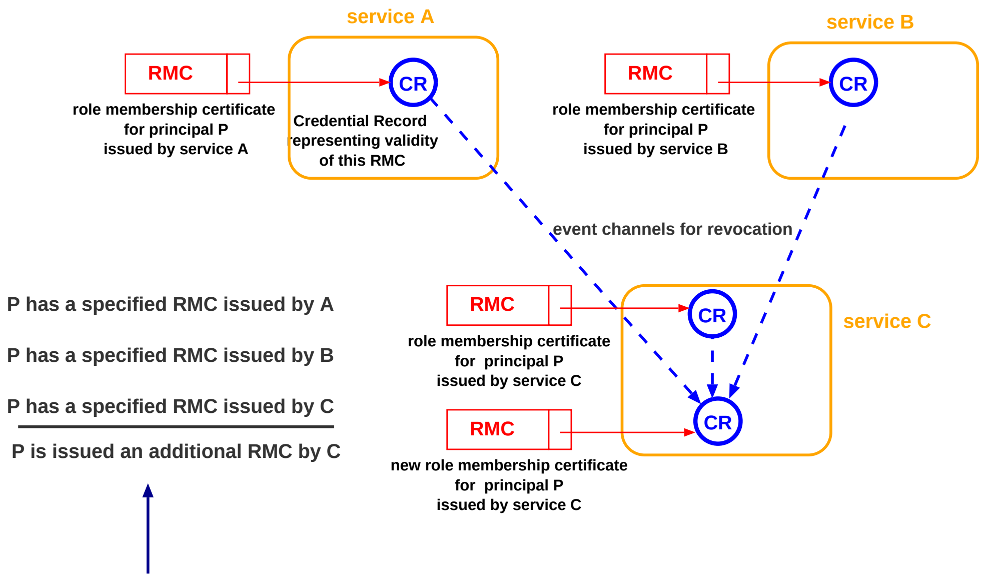
entering a role:

* Oasis services specify policy for role entry
  - in RDL (Role Definition Language)

* Clients present credentials for entering a named role in order to use a service

* Credentials checked against policy specification

* If successful , client is issued a role membership certificate (RMC)
  cf. principal-specific capability



This is the policy specification by service C
of the conditions for principal P to enter some role

It is expressed in a formal language, RDL (Role Definition Language)

then ....

* Client presents RMC with each service use
  - method invocation
  - access to service-managed objects

* Services may be Oasis-aware and require, but not issue, RMCs

OASIS principles:

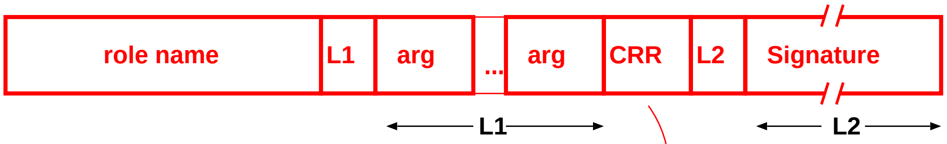
independently developed services

policy expression is per service / per domain

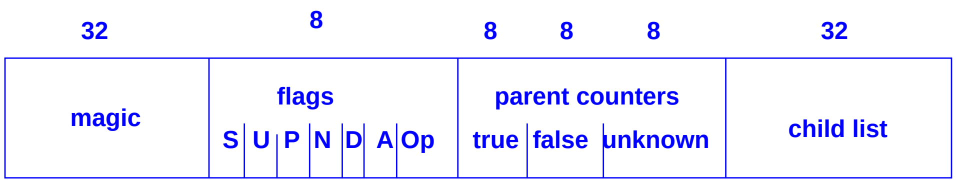
service (domain)-level agreements on use of others' RMCs

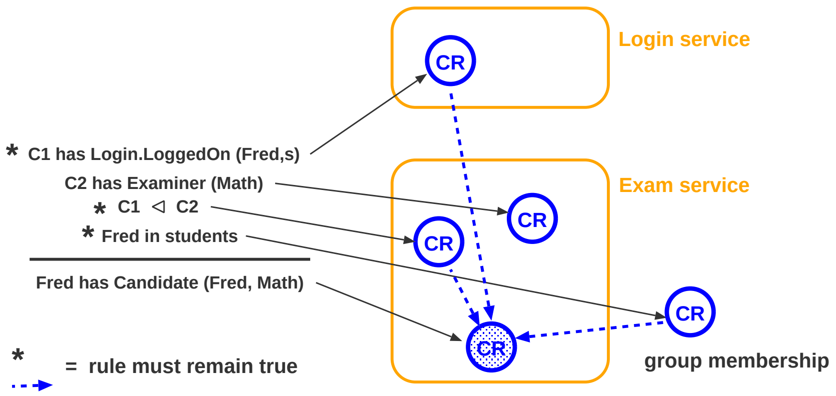
anonymity / privacy via role-name when required

A possible role membership certificate



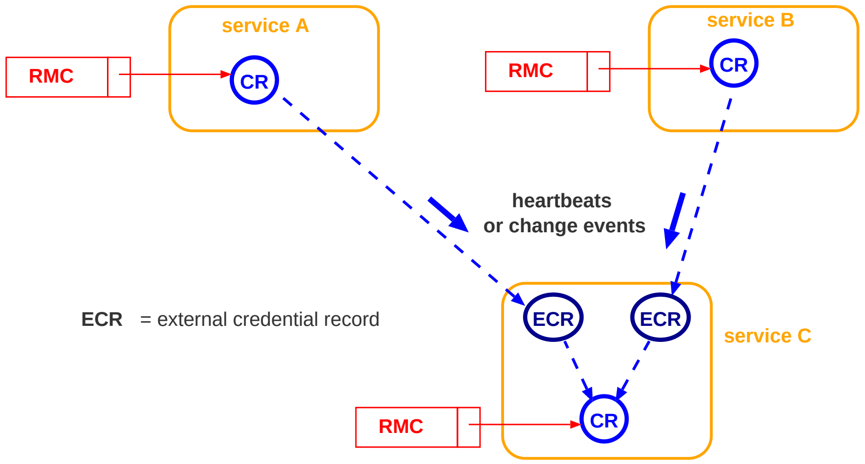
A possible credential record





* C1 ◁ C2 *means that an examiner has issued Fred with a persistent certificate for the role exam-candidate at some time in the past*

* Fred in students *this is shown as a registration database lookup it might also be managed by issuing a persistent student-registration certificate*



OASIS engineering

* independent (service / domain-level) design of RMCs

* service at which RMC is presented checks back
  with issuing service / domain which checks RMC
  (then used-service caches RMC to avoid repeated check-back)

* persistent RMCs for proof of group membership
  embedded principal = user-name

* transient RMCs for dynamic, run-time service use
  embedded principal = process-ID

* Individual, immediate revocation of RMCs
  using event infrastructure