An Architecture for Distributed *OASIS* Services

John H Hine [†]
Walt Yao
Jean Bacon
Ken Moody

[†] Victoria University of Wellington, NZ
Computer Laboratory, Cambridge, UK

Overview – An Architecture for Distributed *OASIS* services

1. background to the research

   people, projects (***EHRs* for the UK *NHS***)

2. fundamentals of ***OASIS***

   - *Role-Based Access Control*
   - *Interoperation of Federated Services*

3. *Certificate Issuing and Authentication servers*

   - *consolidating certificate management*

4. ***Achieving high availability for CIAs***

   - *work of Walt Yao (with John Hine)*

5. *FUTURE WORK*

Experimenting with *OASIS*

Open Architecture for Secure Interworking Services

people

- *OPERA Group* – Computer Lab, Cambridge (UK)
  - ***Jean Bacon***, ***Ken Moody*** (*Faculty*) ***Walt Yao*** (*PhD Student*)
  - see IEEE Computer (March 2000)
- ***John H Hine*** – ***sabbatical visitor, 1999*** – ***principal author, with Walt Yao***
  - Victoria University of Wellington (NZ)

research context

- evaluate *OASIS* against *EHRs* for the UK ***NHS***
  - ***heterogeneous*** *administration* & *software*, *continuous evolution*
  - ***require*** *scalability*, ***high*** *availability*, *secure* audit trail
  - ***work in progress !***

OASIS Access Control *"you've got a ROLE with it . . "* *(pop culture)*

principals (clients?)

- *PERSISTENT* – typically a ***person*** or ***job-title*** – named by ***e.g.*** *NHS_number*
- *TRANSIENT* – a ***computer process*** or ***agent*** – named by ***e.g.*** *unique_process–ID*

scalability of *POLICY expression*

- classify ***clients*** by *ROLE* (parameterised ?), *ROLE* ***names*** *specific to each service*
  - ***e.g.*** *doctor* , *logged-in_user ("Fred")*
  - potential for giving *client anonymity* if required
- specify ***control of access*** in terms of *ROLEs* (of *this* and possibly *other services*)
  - as held by *TRANSIENT PRINCIPALs*
  - *each service* defines its own rules for *ROLE* entry

Long-lived rights for *PERSISTENT PRINCIPALs*

- *APPOINTMENTs* (bound to *PERSISTENT NAMEs*)
  - grant entry to a new *ROLE conditionally on*
    OTHER ROLEs held + *constraints* on their *parameters*
- administered *via* specific *ROLE(s)* (direct expression of *management policy* ?)

Managing *ROLE MEMBERSHIP* and *APPOINTMENT CREDENTIALs*

- via a *signed certificate* ("capability") , format determined by the issuing service
  - *issued to* and *managed by* a ***principal*** , *TRANSIENT* or ***PERSISTENT***
- a *credential record* (maintained at the issuing service)
  - asserts the *validity* of each issued certificate
  - linked to the active conditions for ***ROLE membership***
  - enables *rapid* and *selective revocation*
    + dependent on ***asynchronous notification***

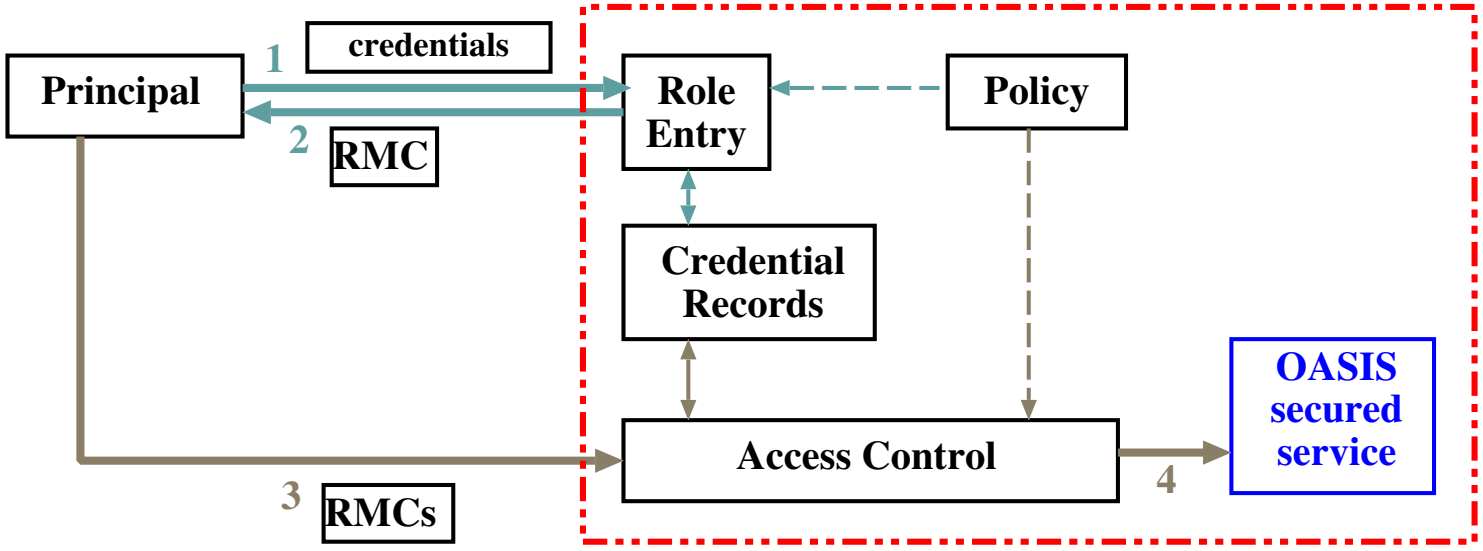
State of *OASIS* implementation

POLICY	application-specific	***more needed !***
ARCHITECTURE	quite detailed	*evaluate . . .*
MECHANISM	certificate based	*evaluate . . .*

Current support for policy expression

- define ***policy*** for a ***service*** by formal languages:
  - *conditions* for *entering* each *ROLE* (***RDL*** , Role Definition Language)
  - *controls* on *METHOD invocation* (use *role parameters* , *method arguments*)
- ***RDL*** Horn clause logic – ***conjunction*** *of conditions* for *ROLE entry*
  - *create a new Credential Record* for *each certificate* issued
  - *maintain the Credential Record graph* , *linking CRs by pre-condition*
  - *links* exhibit ***certificate dependencies***, so ***explicit trust*** between services

A service secured by OASIS access control

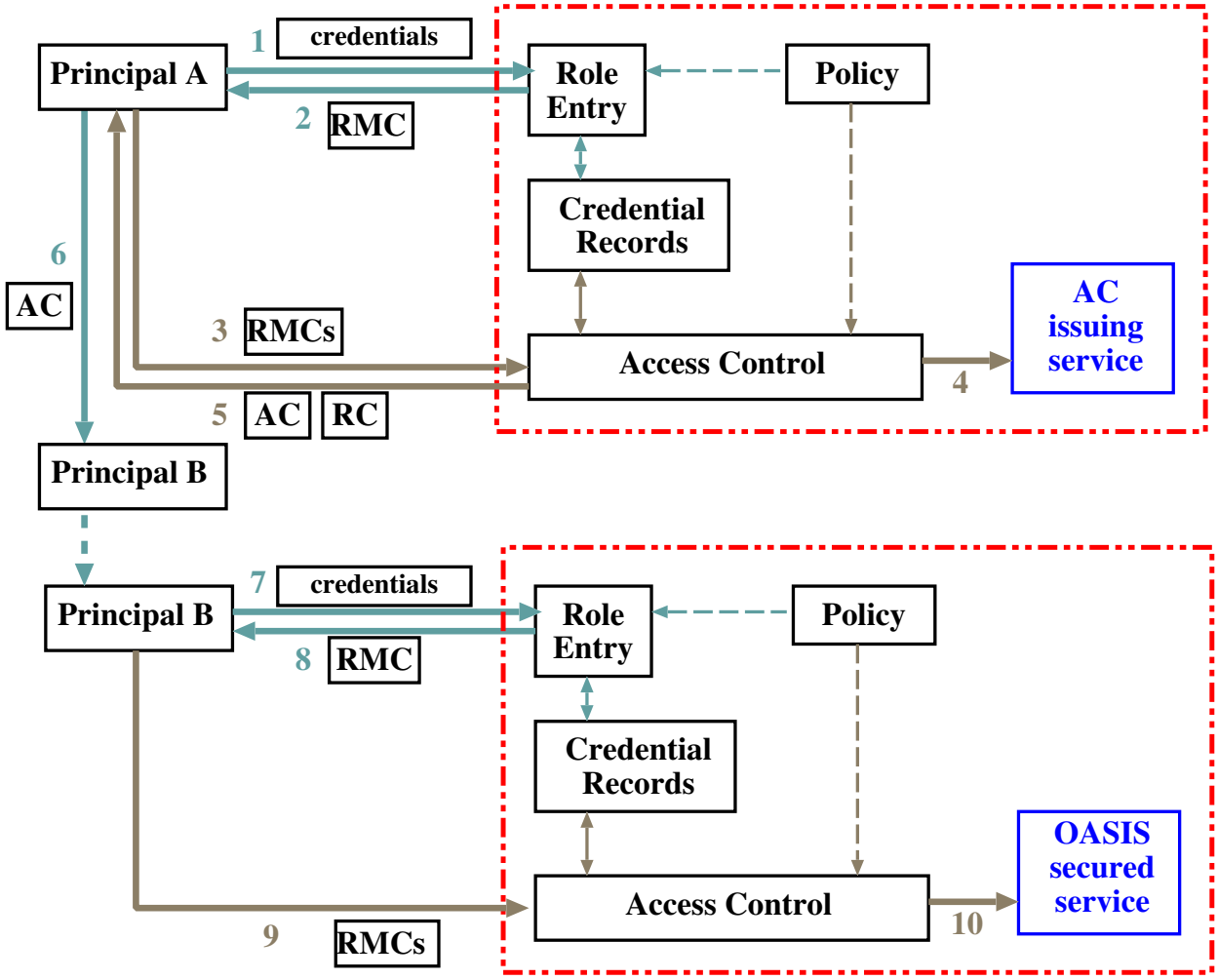


RMC = role membership certificate

→ = role entry

→ = use of service

Issuing and Using Appointment Certificates



1. principal A enters role *AC-issuer*
2. RMC as *AC-issuer* returned
3. *AC-issuer* requests an AC for principal B
4. validated request passed on
5. AC and RC returned to principal A
6. principal A passes AC to principal B but keeps RC

. . . . . .

7. principal B enters a role using AC as one credential
8. RMC returned to principal B
9, 10. standard use of OASIS secured service

RMC = role membership certificate, AC = appointment certificate, RC = revocation certificate

→ = obtaining and using credentials for role entry

→ = use of service

Certificate Issuing and Authentication

Considerations governing certificate management

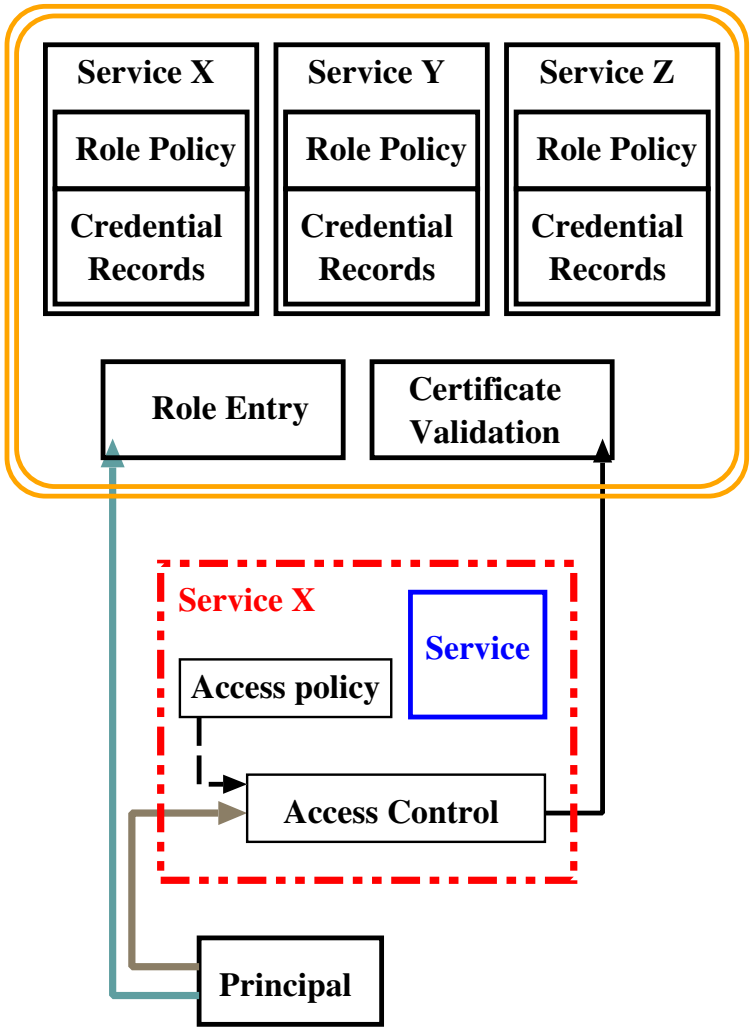
- separate ***ROLE entry*** (***OASIS issuing service***) , ***ROLE use*** (***OASIS aware service***)
- protecting certificate management is ***critical*** to the ***effectiveness of security***
- certificate validation on ***ROLE entry*** requires ***authenticated communication***
- replication of ***CIA*** servers can improve both ***availability*** and ***responsiveness***

What criteria determine pooling of CIA function ?

- administrative responsibility for ***policy expression*** covering different ***services***
- clustering of ***trust relationships*** and ***certificate exchange*** between ***services***
- administrative overhead of ***secure server management***

(single replicated CIA service within a hospital ?)

Certificate Issuing and Authentication (CIA) Service



CIA service (reliable, replicated)

Oasis-secured service

Managing replication of CIA function

General strategy for high availability

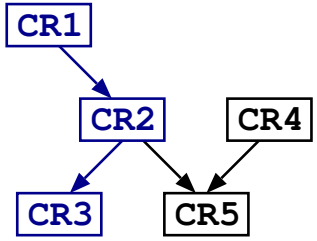
- keep a full replica of the local ***CR graph*** at each ***CIA server*** of the local federation
- all available ***CIA servers*** of the local federation offer an "identical" service to clients
- maintain weak consistency of replica ***CR graphs*** using an ***update-notify*** protocol
- ***failure detection*** and ***recovery*** protocols make the local ***CIA*** federation fault tolerant

Assumptions regarding replica management

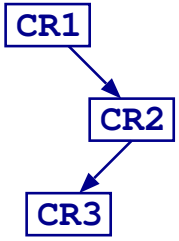
- two forms of ***update*** action : ***add_CR (list-of-links)*** , ***remove_CR***
- changes to ***CR graphs*** are made ***atomically***
- ***update-notify*** messages are ***broadcast reliably*** to all available ***CIA servers***
- ***CRUCIAL!*** (see details in the paper) : ***update*** actions on ***CR graphs*** are
  - ***individually idempotent*** and ***pairwise commutative***

Order Independence of Concurrent Updates

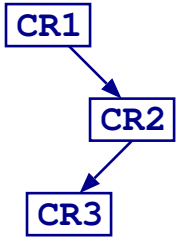
CIA Server A



(a) Initial state

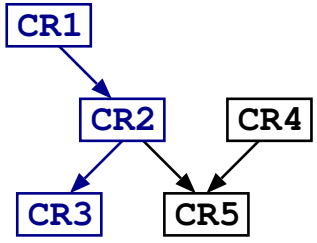


(b) CR4 is revoked

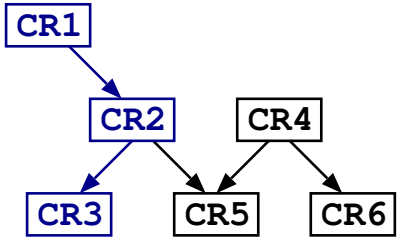


(c) State consolidated

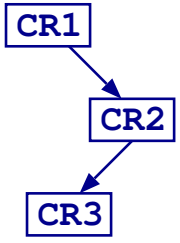
CIA Server B



(a) Initial state



(b) CR6 is added under CR4



(c) State consolidated

General model for ensuring CIA recovery

5-state operation at each replica server of a federation

- ***NORMAL*** "***all peer servers alive***" exchange ***updates*** by ***reliable broadcast***
- ***Replay Logging*** "***at least 1 peer has failed*** " maintain a ***structured update log***
- ***DOWN !***
- ***RECOVERING*** "***this replica is restoring state***" ensure local ***CR graph*** is current
- ***Coordinating*** "***assisting a peer to recover***" deliver ***updates*** lost while ***DOWN***

Tactics for bringing the local CR graph up to date

- note the ***transience*** of ***ROLE certificates*** , ***high volatility*** of the ***CR graph***
- short-term problem**:** ***replay*** the ***structured update log*** with help of a ***coordinator***
- longer-term failure**:** ***synchronized state transfer*** with help of a ***coordinator***
- ***no need for causal ordering of update messages !***

Current and future work

Problems specific to CIA server federation

- more formal analysis of the ***replication protocol***
- setting up a ***threat model*** for failure and recovery
- separate treatment of ***persistent*** and ***transient*** ***CRs***

Deployment and Evaluation of *OASIS*

- ***Eastern Region EHR Consortium*** (for UK ***NHS***)
- experimental population base of about 7 million
- many partners (including IBM Hursley, MQ team)
- encouragement from ***NHS*** planners but no funding as yet . . .
- ***people*** & ***policy*** problems (intimidating)