Modular Authorisation Infrastructures

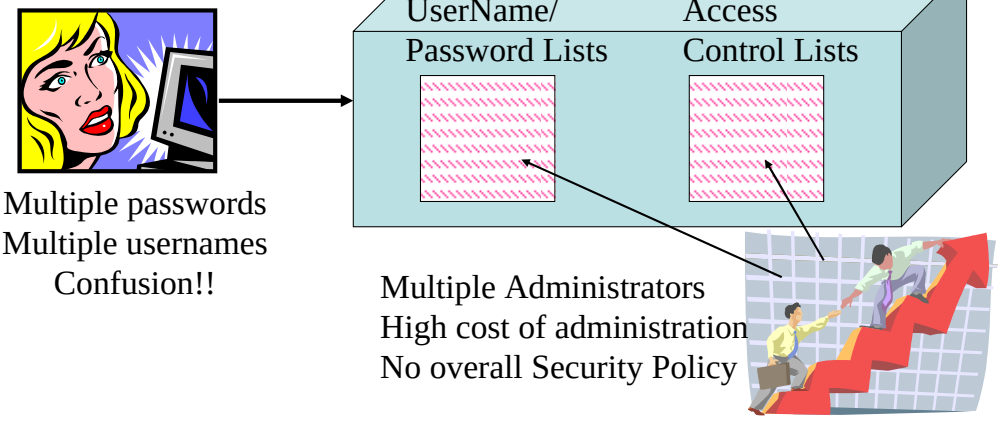
David Chadwick
University of Kent

Contents

- An introduction to authorisation and access control models
- DAC, MAC and RBAC with attribute certificates
- Trust/Authorisation Management ala Keynote
- ISO 10181-3 Access Control Framework
- XACML
- Grids and VO Distributed Authorisation
- Some more complex requirements
  - Break the Glass Policies
  - Support for Multiple PDPs executing different policies
  - Coordinated Decision Making
  - Level of Authentication/Assurance (LOA)
  - Support for Sticky Policies
- Building application independent PEPs and obligation services

Traditional Applications

- Authentication and Authorisation are Internal to the Application
- Typically based on weak passwords



Multiple passwords
Multiple usernames
Confusion!!

Multiple Administrators
High cost of administration
No overall Security Policy

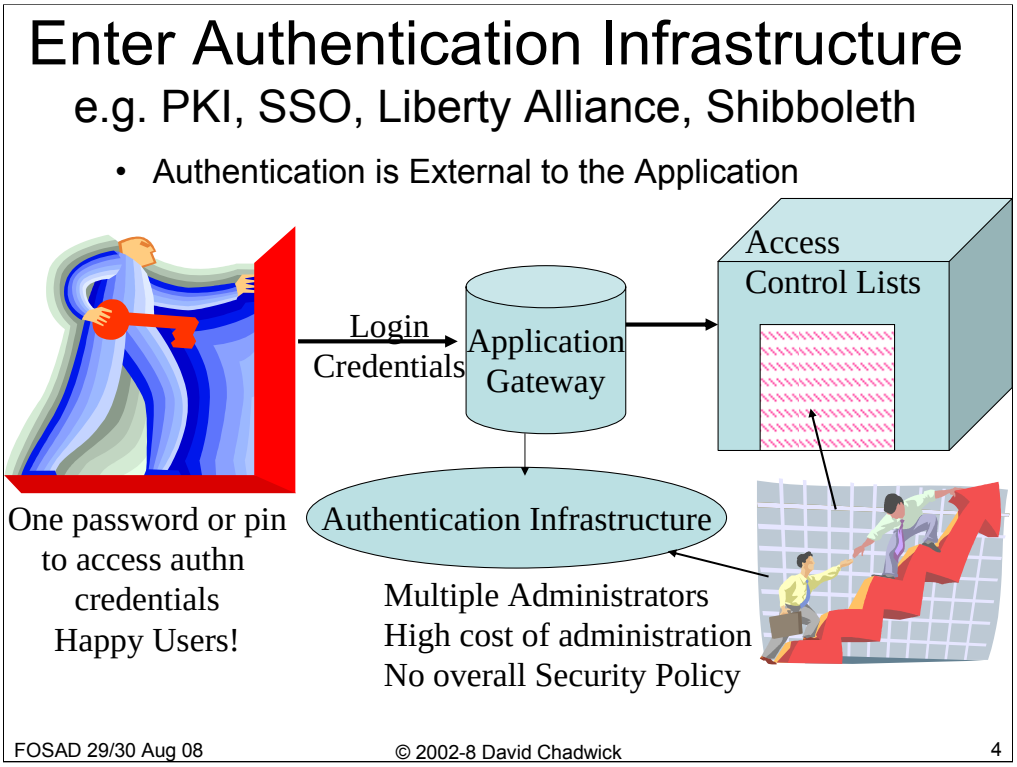
FOSAD 29/30 Aug 08 © 2002-8 David Chadwick 3

In traditional applications:

Authentication information is held in username password lists

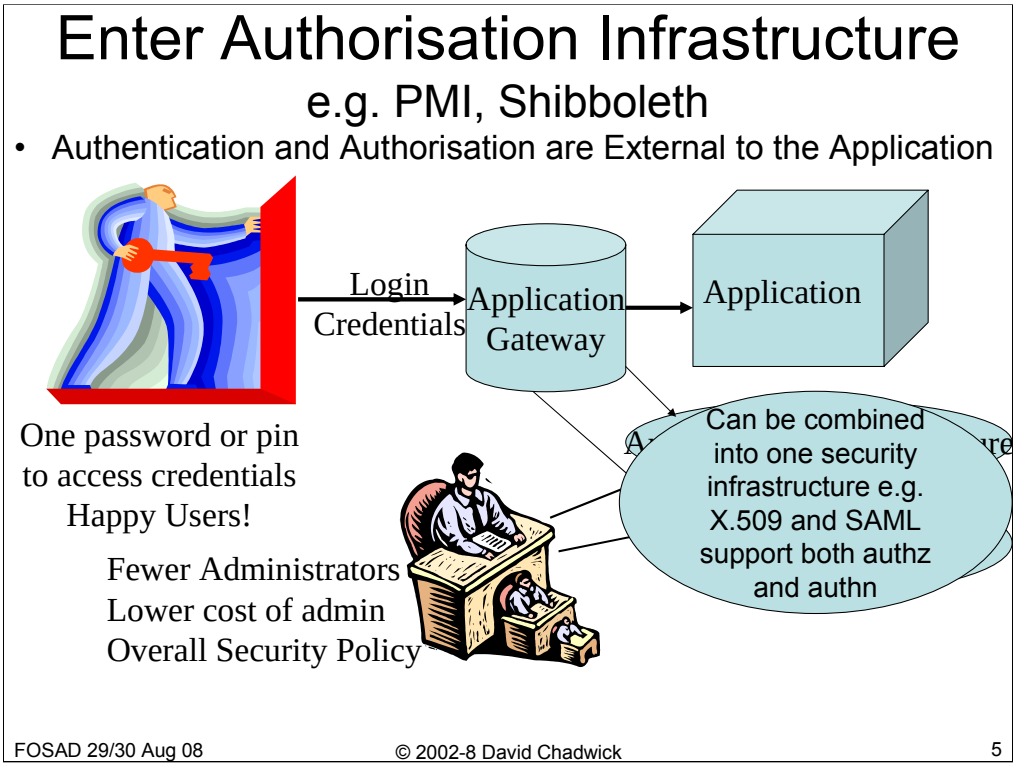
Authorisation information is held in access control lists.

This leads to high administration costs and users having to have multiple usernames and passwords. It is also impossible to have a security policy that is enforced by the multiple applications in the organization.



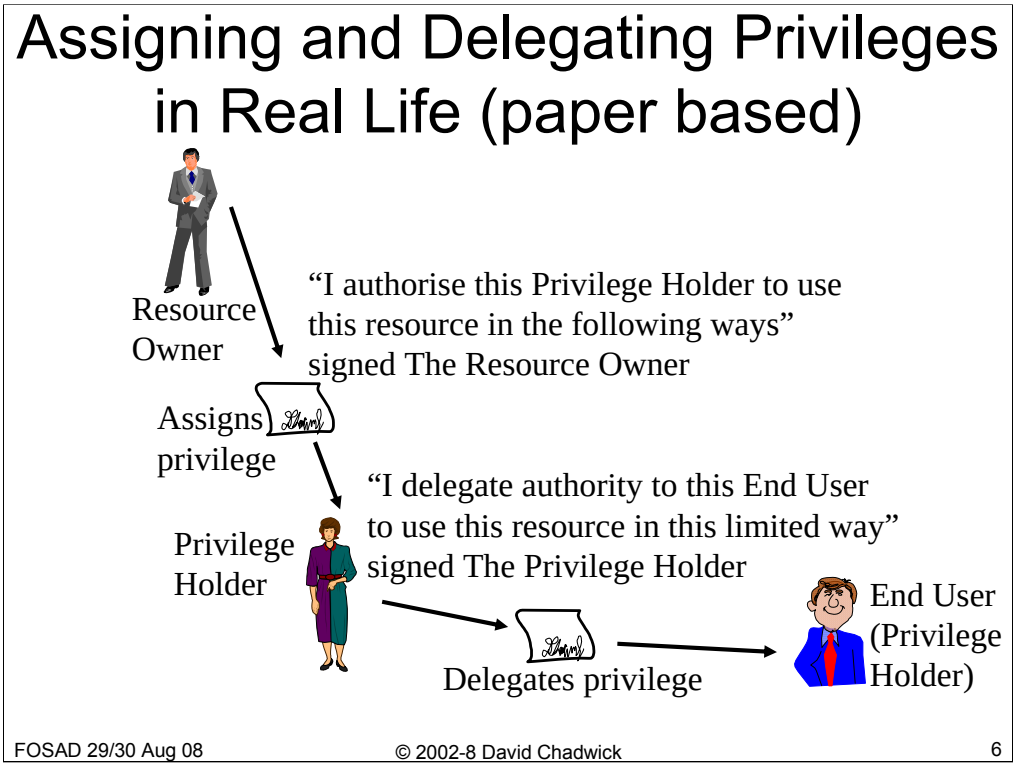
By removing the authentication function from all applications and giving it to an outside entity, we can provide the users with single sign on. PKIs with application gateways that check the user's signatures, can be one way of providing single sign on. The users are much happier when they have single sign on, since they only need to know a single password to access their credentials.

However, single sign on does not usually provide common administration of authorisation, so there is still a high cost here of administering multiple access control lists.



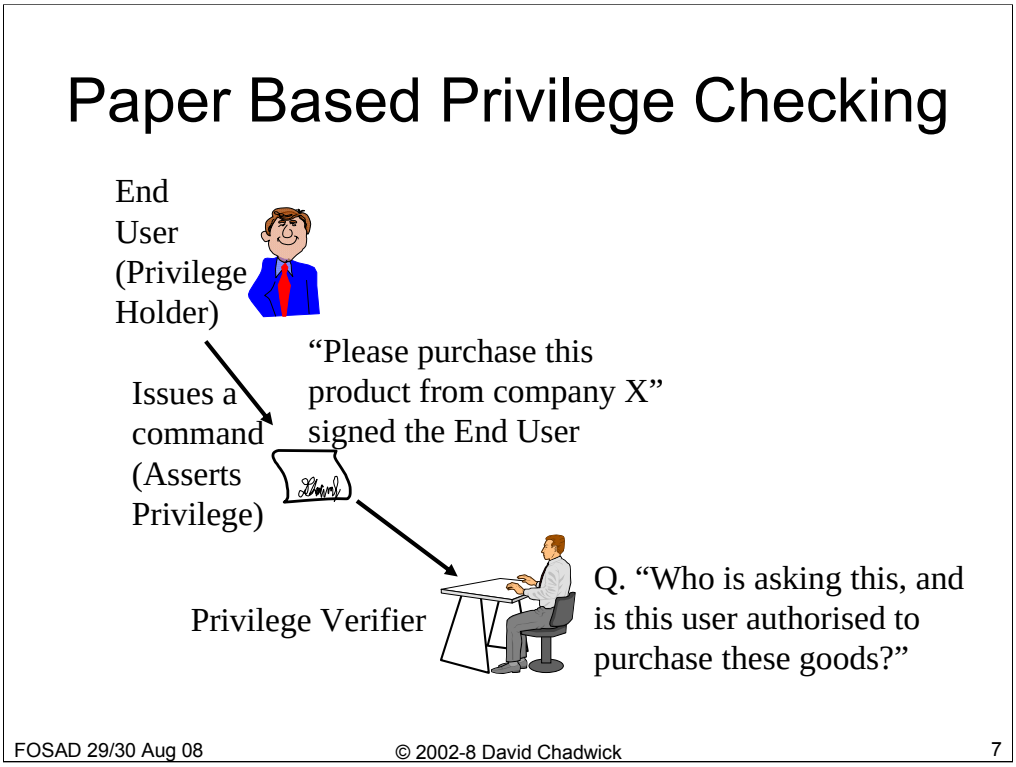
If we also remove the authorisation function from the applications, and provide this in an external server, we can now benefit from single authorisation management, and govern all authorisation decisions through a common security policy. This makes administration of authorisation much easier and lowers its overall cost whilst simultaneously increasing the overall security.

When a user leaves an organisation, we now only need to record this fact in a single point (usually the authentication infrastructure) and the user is then barred from accessing all applications.



In a paper based privilege management system, a resource owner (e.g. the Financial Director of a company, or the Computer Centre Manager) will sign a form to say that a particular person (the privilege holder) is allowed to use a particular resource in a particular way. E.g. The Financial Director may say that a Head of Department can sign orders up to the value of so many thousand Euros, or the Computer Centre Manager may sign a form authorising a user to use particular computing resources.

Paper based systems may also support delegation, whereby a privilege holder is allowed to delegate the use of the resources currently under his control, to one or more other people. E.g. the Head of Department authorises a project manager to sign orders for his project up to a pre-determined sum.



When the privilege is asserted (or exercised) by the privilege holder, the privilege verifier needs to be sure that the privilege really does belong to the user claiming the privileges.

E.g. a purchase order clerk (privilege verifier) will need to check such things as

i) is this user allowed to sign Internal Requisitions

ii) who gave permission for this user to sign Internal Requisitions

iii) is this Internal Requisition within the limits of what he is allowed to purchase

iv)are all the signatures valid or are they forgeries

v) has the user's privilege been withdrawn since the last time I did the checking

Making this process electronic

- The following (at least) are needed for a PMI:
- Electronic tokens (authz credentials) to hold and carry user privileges
- Protocols for passing these credentials around
- Authz decision engine that can answer the question "Is this user allowed to do this task?"
- Policy that provides the rules for the decision engine
- Audit trails/logs that keep a tamperproof record of user requests and responses
- GUIs to allow users and administrators to interface with the system
  - assign privileges to users, revoke privileges from users
  - set policies, modify policies, view/search audit trails
- Application code that will perform the user's actions

FOSAD 29/30 Aug 08 8

Privilege Management Infrastructure (PMI) is the term used in the X.509 standard for the authorisation infrastructure, in order to distinguish it from the PKI (public key infrastructure) authentication infrastructure.

Authz Credentials (Attribute Certificates/Assertions)

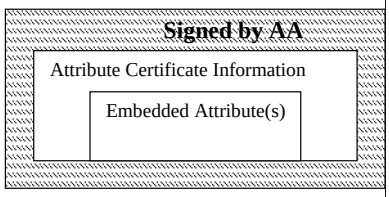
- All attribute certificates/assertions should contain
- Identifier of the holder
- Identifier of the issuer
- Validity period of the certificate
- The set of attributes being bound to the holder
- Policy information of the Issuer
  - restrict its use, restrict issuer's liabilities, help RP determine its validity etc.
- Unique number of the certificate
- Signature of the Issuer and algorithm identifiers
- Two standards for authz credentials: SAML attribute assertions and X.509 attribute certificates (or public key certificates)

FOSAD 29/30 Aug 08 9

It does not matter how the certificates are actually encoded e.g. ASN.1 DER, XML etc. The syntax is not that important.

X.509 Attribute Certificates

- Comprises a binary SIGNED SEQUENCE of:
  - version number of this AC (v1 or v2)
  - the unique serial number of this AC
  - the holder
  - the issuer
  - the identifier of the algorithm used to sign this AC
  - the validity period of this AC
  - the sequence of attributes being bound to the holder
  - any optional extensions
- created using ASN.1 binary encoding rules



FOSAD 29/30 Aug 08 10

The attribute certificate is used to hold a user's privileges

SAML Attribute Assertion

- Comprises an XML character string comprising:
- Version Number (currently 2.0)
- Unique ID of this Assertion
- Time of Issuance
- Issuer
- Subject (optional)
- Signature (optional)
- Optional Conditions, which include validity time
- Optional Advice (which can be ignored)
- The attribute statements about the subject

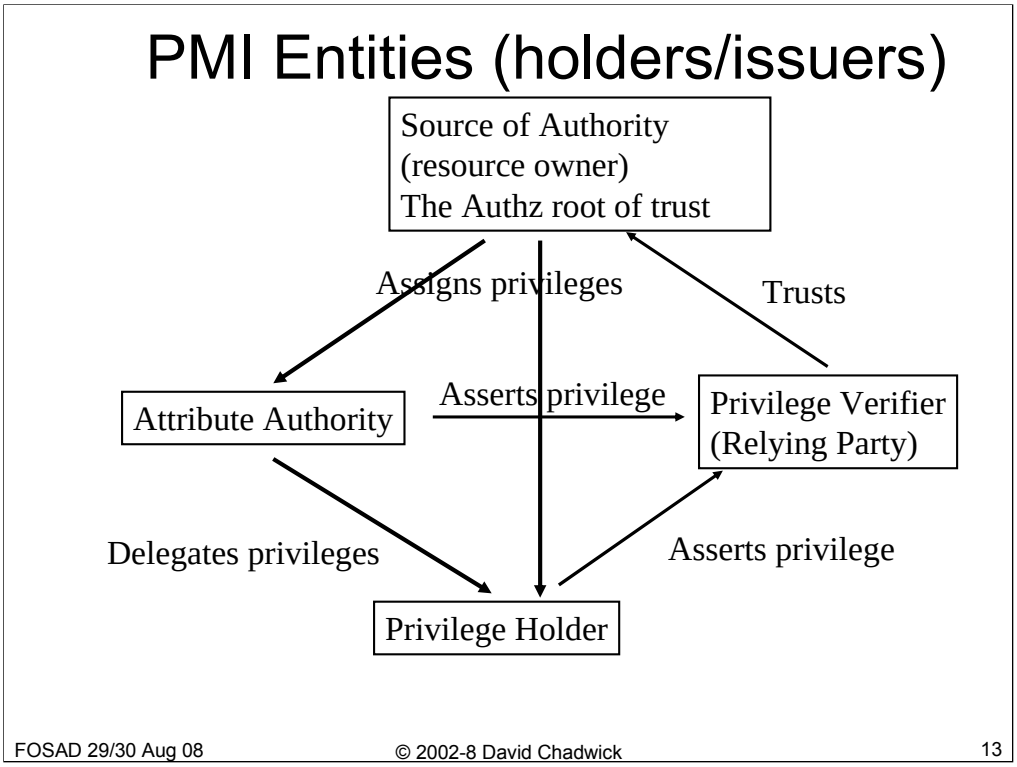
Differences between X.509 ACs and SAML Attribute Assertions

• Encoded in binary (DER)	• Encoded in XML text
• Mandatory signature	• Optional signature
• Can be short or long lived	• Must be short lived
• Can be revoked	• Cannot be revoked
• Can be delegated	• V1 cannot V2 can
• Mandatory validity period	• Optional validity period
• No time of issuance	• Time of issuance
• Serial number	• Unique Identifier
• Optional critical extensions	• Optional conditions
• Optional non-critical extensions	• Optional advice

- So they are very similar

FOSAD 29/30 Aug 08 © 2002-8 David Chadwick 12

In X.509 time of issuance was not seen to be important, but rather the validity time period was seen to be more important, especially for long lived credentials. (You don't really care when a plastic credit card is created, only when it can be used – the validity time). For short lived credentials that are issued on demand, you want to know when it was issued since this is usually the start of the validity period, although in SAML they can be different.



The X.509 Privilege Management Infrastructure is very similar to the existing paper based privilege management infrastructure, and there is direct correspondence between the entities.

The Source of Authority is the Resource Owner and the root Attribute Authority. The SOA assigns privileges to other entities - Attribute Authorities and Privilege Holders.

A privilege holder is an entity that has been assigned a privilege, and it may assert the privilege but is not allowed to delegate the privilege to anyone else.

An attribute authority is an entity assigned privileges, that is allowed to delegate privileges to other entities (attribute authorities and privilege holders). An attribute authority may or may not be able to assert the privileges itself.

The privilege verifier is the relying party that checks the asserted privileges and makes a yes/no decision as to whether the privilege may be used or not. The privilege verifier trusts the source of authority (SOA) as the trusted issuer and checks that the privilege holder has been authorised by the SOA or one of its delegated AAs.

Implementing Access Control Models with Attribute Certificates/Assertions

- The different access control models are
  - Discretionary Access Control (DAC)
  - Role Based Access Control (RBAC) or more generally Attribute Based Access Control (ABAC)
  - Mandatory Access Control (MAC) or Multi-level Security (MLS)

Discretionary Access Controls

- Users may optionally be given access to resources by the resource holder
- The privileges are usually held in Access Control Lists in the Resource
- Either user first or privilege first lists

User1	r, w, e, d
User2	r, e
User3,4	r

r	User3,4
r, e	User2
r, w, e, d	User1

FOSAD 29/30 Aug 08 15

Everyone is familiar with the traditional Discretionary Access Control model, as this is typically implemented as access control lists in filestores, file serves, databases, operating systems etc. The access control lists are actually held within the resource itself, which makes it fast to compute the access control decisions. The scheme is also easy to understand, but has the disadvantage that when there are large numbers of users it can be complex to manage. Users have to be given access rights (privileges) to each resource they want to access, and when they leave or change jobs, they have to have their permissions removed from each resource (which isn't always done!). If the privileges were held external to the resource, then multiple resources could use the same privileges, which would make the management a lot easier.

DAC with Attribute Certificates

- The user (holder) is given an Attribute Certificate which strongly binds his/her identity to the privileges being given to him/her
- The AC is signed by an Attribute Authority (Resource Owner or his delegate)
- Similar to X.509v3 PKC, only it holds a sequence of attributes rather than a public key
  - Each attribute is one privilege e.g. read File X
- An attribute certificate/signed attribute assertion can be transported and stored anywhere since it is secure and self contained

Attribute certificates/signed attribute assertions can be used to hold authorisation information (privileges) in much the same way the public key certificates hold authentication information. A significant difference between a PKC and an AC is that only a CA can issue a public key certificate (and this is a specialised function and they are usually very limited in number), but anyone holding privileges can be allowed to act as an AA and issue attribute certificates to others, in order to delegate privileges to them (these are not such a specialised function and might be very many in number).

Mandatory Access Controls (MLS)

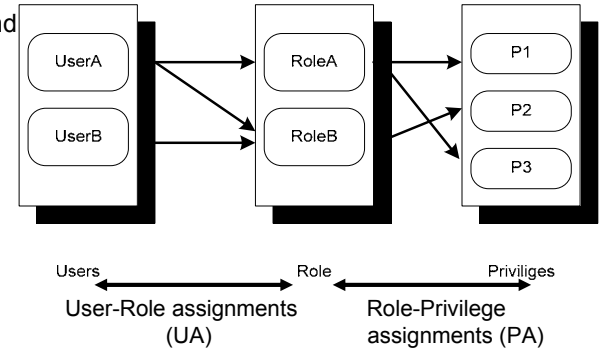
- Every target is given a security label which includes a classification
  - E.g. unmarked, unclassified, restricted, confidential, secret, and top secret
- Every subject is given a clearance which includes a list of classifications they can access
- Security policy links the two
  - E.g. Read down, write up policy stops info leakage
  - Subject who logs in as restricted can read unmarked -> restricted and write restricted->confidential

MAC with Attribute Certificates

- Subjects are given clearance attribute certificates
- The privilege attribute in the AC now holds the users clearance
- Targets can be securely configured with their own security label (also as an AC)

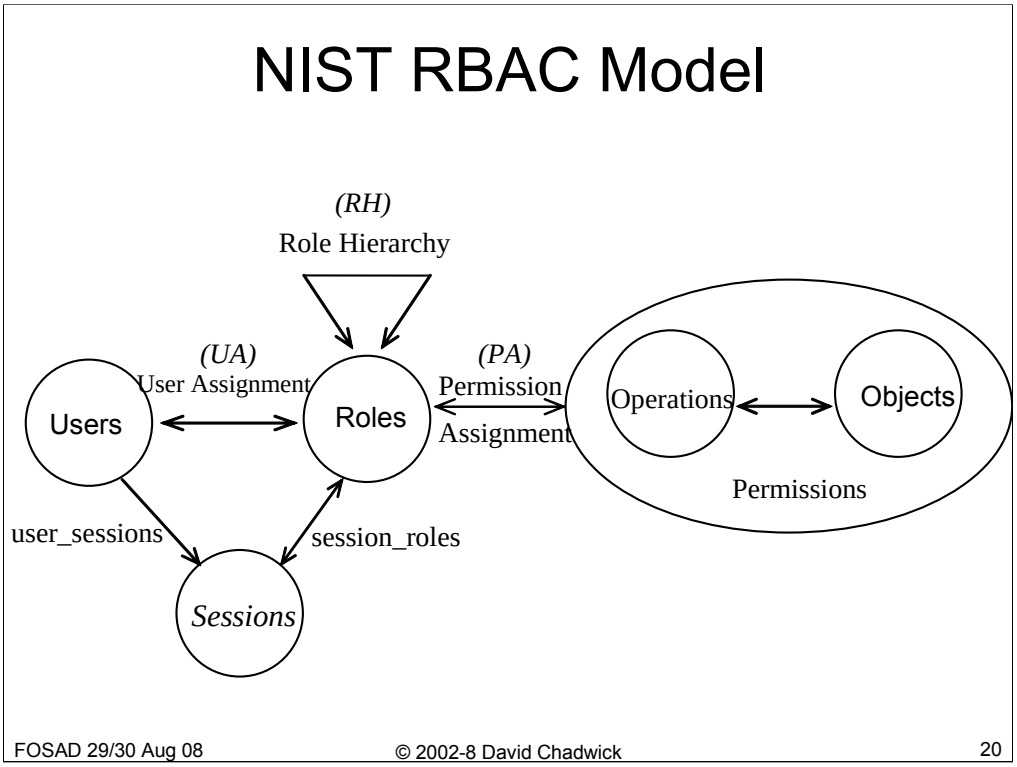
Role/Attribute Based Access Control Model

- Hierarchical Role based Access Control (RBAC)
  - Permissions are allocated to roles/attributes
  - Superior roles/attributes inherit privileges of subordinate roles/attributes
  - Users are assigned role memberships
  - Role members acquire roles' permissions
- Benefits
  - Security
    Remove a user's roles and all privileges are gone
  - Manageability
    Users change more frequently than roles
  - Scalability
    No of roles usually much less than no of users



FOSAD 29/30 Aug 08 © 2002-8 David Chadwick 19

The main difference between roles and attributes is that in general, roles can be delegated (e.g. the role of project manager) but attributes cannot be delegated (e.g. a person's age, or organisation they work for).



RBAC has been standardised by ANSI as an American Standard, from work developed by NIST.

RBAC with Attribute Certificates

- Role Specification Attribute Certificates assign privileges to roles (the holder is a role name). These are the Permission Assignments of NIST
- Role Assignment Attribute Certificates assign roles to people, using the role attribute. These are the User Assignments of NIST
- The role membership and role privileges can be separately administered if wanted

FOSAD 29/30 Aug 08 21

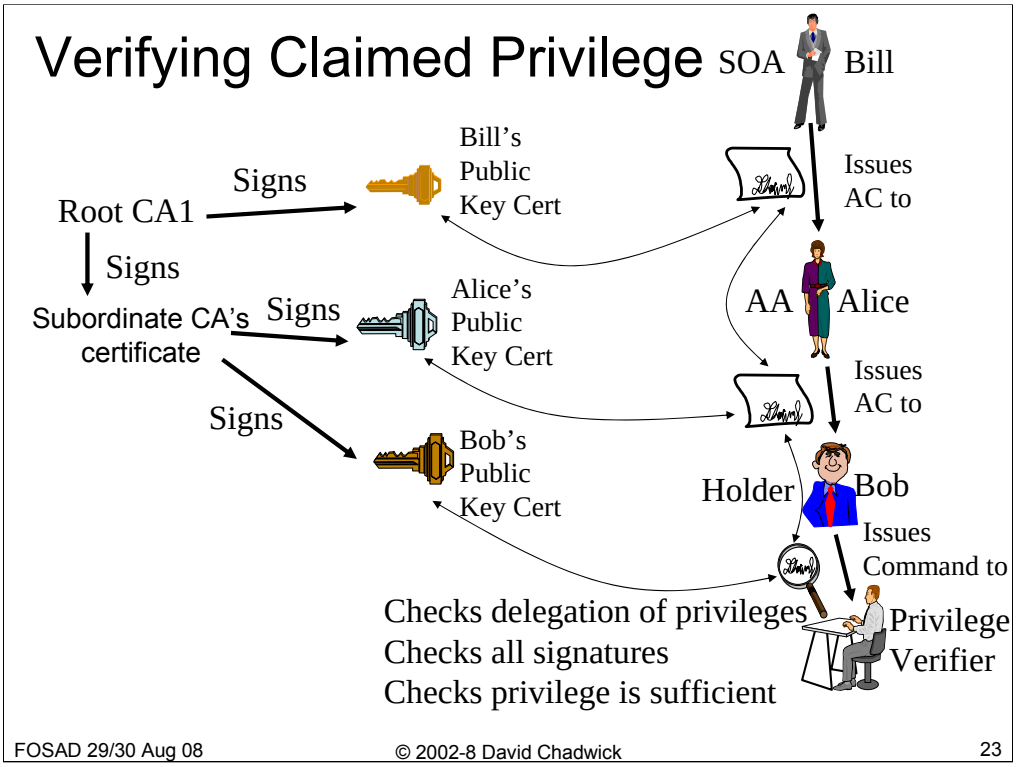
Role Specification Certificates are a special type of attribute certificate, that define the privileges that go with a role. The holder of the certificate is a role name rather than the name of a person. Further, role specification certificates cannot be delegated (as can normal attribute certificates).

Roles are given to people in role assignment attribute certificate, where the attribute is the role attribute. Role memberships can be delegated if wanted.

The management of the privileges assigned to a role, and the membership of the role can be administered separately by different AAs if this is needed by an organisation. If this is the case, the role attribute can indicate which AA is responsible for allocating privileges to the role.

Conclusion

- Any of the 3 well known access control models can be implemented using attribute certificates/assertions instead of storing the access control data internal to the application
- Once we have externalised the access control data, we can also externalise the processing of it to build a modular application independent authorisation infrastructure
- But processing this data may be complex!



Verifying a delegated chain of attribute certificates is a complex process that requires not only validating the chain of attribute certificates (e.g. Bill to Alice to Bob), but for each attribute certificate in the chain (e.g. Alice's AC), validating the chain of public key certificates that were used to sign this attribute certificate,

Verifying the Claimed Privilege

- Check AC chain is valid back to SOA
  - Signer is an AA, and delegation rules have been adhered to
  - No Attribute Certificate has been revoked
- Check all public keys are valid
  - Requires checking public key certificates and CRLs back to all the root CA public keys via any intermediate CAs
- Check privilege can be exercised now on this resource
  - Time, service, and policy restrictions are being obeyed
- Check privilege is sufficient for access method
  - Check against policy and local variables
- Check holder wanted to exercise this privilege
  - E.g. signed request contains pointer to AC

FOSAD 29/30 Aug 08 24

Verification is a complex process and involves:

- validating the AC chain back to the SOA, and determining that no AC in the chain has been revoked, that each AA in the chain was allowed to delegate privileges and that the privileges were properly delegated,

- validating all the public keys used to sign the AC chain and the request to access the resource. This includes retrieval of all the latest CRLs,

- checking that the claimed privilege is valid for this resource at this time, by checking any time restrictions, service restrictions and the policy in force

- checking the privilege against the policy to see if it is sufficient for the mode of access being requested

- checking the privilege against any local variables such as time of day, or credit balances to see that these are not being exceeded,

- and finally checking that the privilege holder did actually want to exercise the claimed privilege. How this is done is not specified in the standard, but one suggestion is that the request from the privilege holder to the privilege verifier is signed, and contains a pointer to the AC being exercised e.g. its AA name and serial number, or the request contains the AC in full.

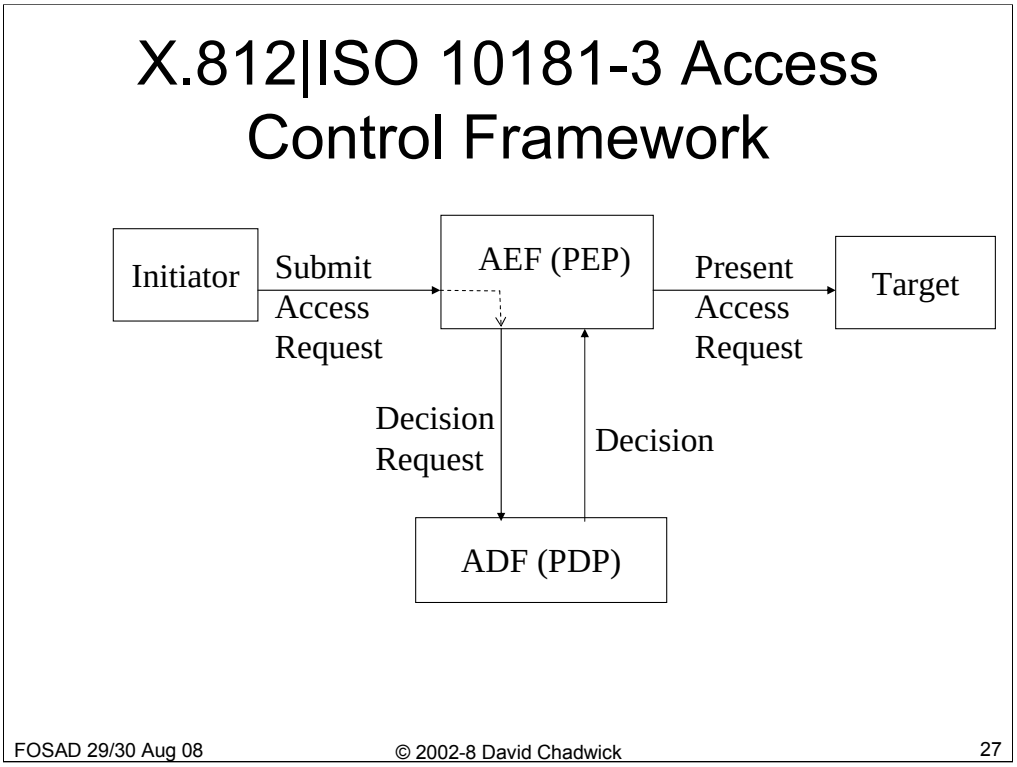
Trust/Authorisation Management ala Keynote

- Blaze in RFC 2704 states five components are needed for trust management (PMI)
  - A language for describing actions (operations)
  - A mechanism for identifying principals (users and resources)
  - A language for specifying application policies
  - A language for specifying credentials
  - A compliance checker which determines how an action requested by a principal should be handled, given a policy and a set of credentials

RFC 2704 The KeyNote Trust-Management System Version 2. M. Blaze, J. Feigenbaum, J. Ioannidis, A. Keromytis. September 1999.

What standards are available for the various PMI components?

- A language for describing actions/operations on resources
  - XACML to a certain extent, but still requires application specific definitions of specific actions and their parameters.
- A mechanism for identifying principals (subjects)
  - X.509 PKCs, SAML, XACML
- A language for specifying application policies
  - XACML
- A language for specifying credentials
  - X.509 ACs and SAML attribute assertions
- A compliance checker
  - ISO 10181-3 model and XACML model



ISO 10181 specifies a standard framework for controlling access to target resources. The access control function is split into two components, the Access Control Enforcement Function (AEF) and the Access Control Decision Function (ADF). The former is responsible for enforcing the access controls, whilst the latter is responsible for making the decision about whether access can be granted or not.

The initiator (which may be a human or an application) submits an access request to the target. The Access Control Enforcement Function ensures that only allowable accesses, as determined by the Access Control Decision Function, are performed on the target.

The AEF will normally be responsible for authenticating the initiator, and only if authentication is successful, will it call the ADF.

In traditional systems the AEF and ADF are all part of the same target process.

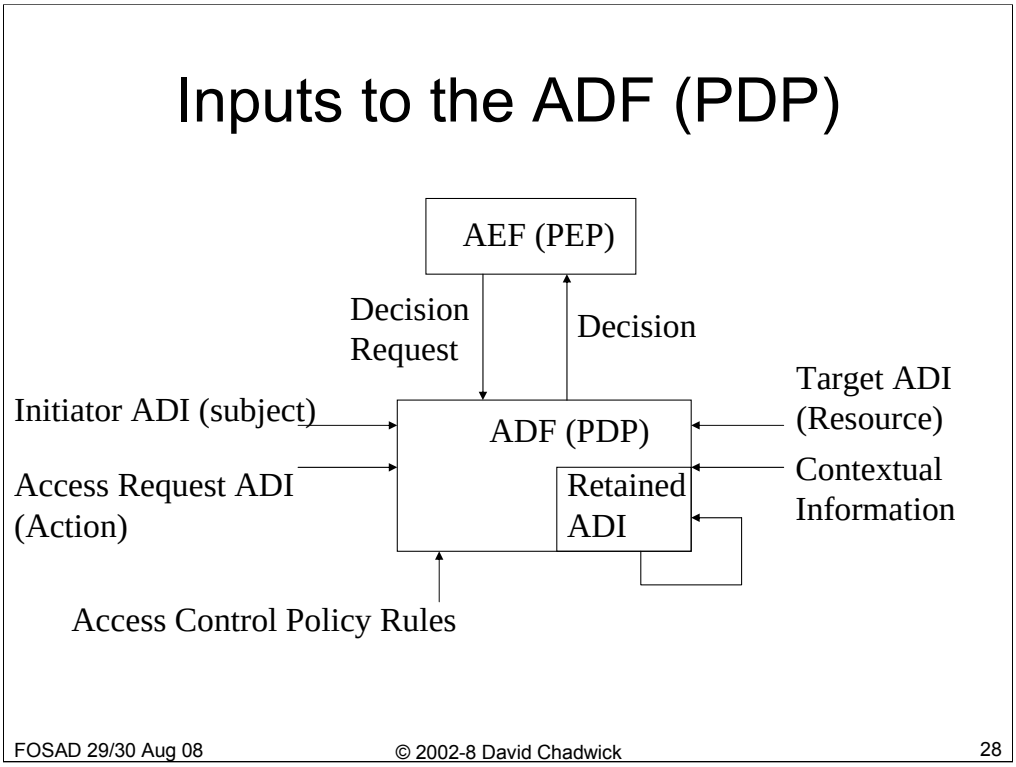
AEF – is the application dependent Access control enforcement function, This knows how to talk the specific application protocol to communicate with the user (initiator) and the target resource

PEP – Policy enforcement point, alternate name for AEF (came from RFC 2904)

ADF – access control decision function. This is the application independent policy controlled decision engine

PDP – policy decision point, alternate name for ADF (came from RFC 2904)

J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, D. Spence. “AAA Authorization Framework”. RFC 2904 August 2000



ADI is the access control decision information that is needed by the ADF/PDP policy engine in order to make its decision. In order to make its decision, the ADF needs access control decision information (ADI) about the following:

- the initiator

- the target

- the access request

as well as requiring the access control policy rules and contextual information e.g. the time of day.

Examples of Access Control Decision Information are as follows:

Initiator ADI - the authenticated name of the initiator, or the role of the initiator.

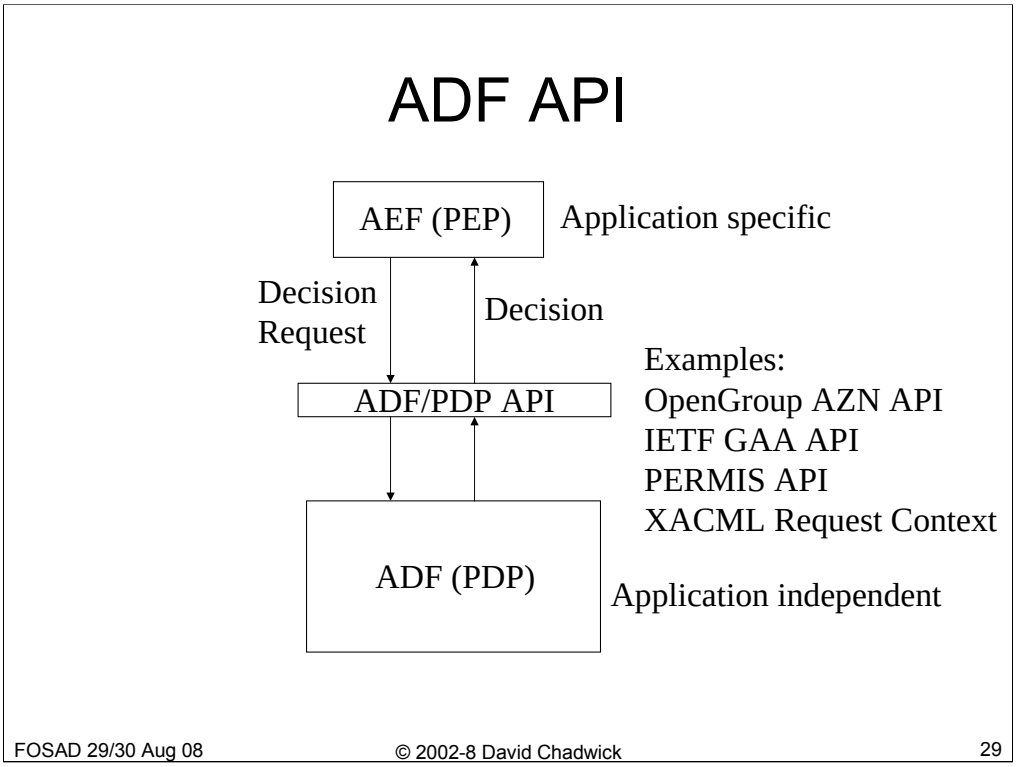
Access Request ADI - the requested operation

Target ADI - the identity of the target or sensitivity markings of the target.

Sometimes access decisions are based on previous access decisions, and so the ADF may retain the ADI from previous requests for use in subsequent decisions. An example of this might be that a file can only be deleted after two different people have retrieved its contents.

In traditional systems the AEF and ADF are part of the same target process, and the access control lists form the access control policy rules.

In X.509 PMI based systems the AEF and ADF may be distributed from the target system, and may even be used by multiple targets.



When policy based access controls are implemented, the ADF needs to have access to the policy. It would be useful if one policy and one ADF could be used by multiple targets. It would also be useful if you could build an application without having to re-implement the ADF each time.

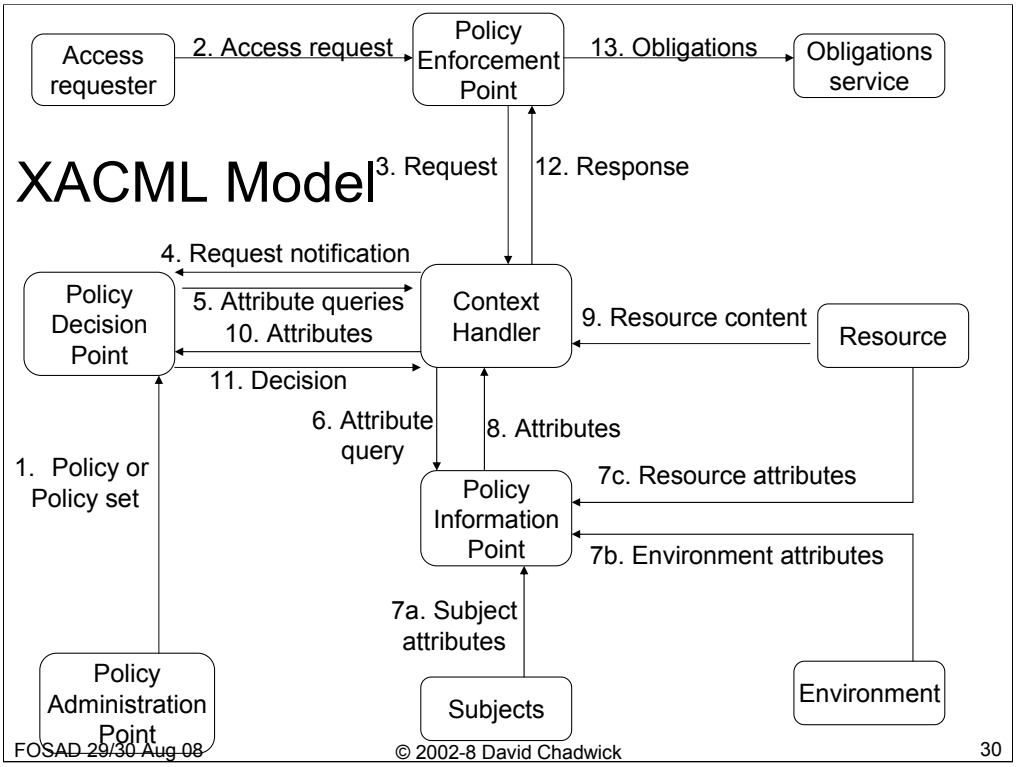
For these reason work is on going to standardise the API between the AEF and the ADF.

The OpenGroup's Authorization API (AZN) API is the first such standard to be published, in January 2000, ISBN 1-85912-266-3. The API is specified in the C language.

The IETF CAT group also worked on the Generic Authorization and Access control Application Program Interface (GAA API) <draft-ietf-cat-acc-cntrl-frmw-0n.txt> where n is the draft number, currently at 6. The GAA API is language independent, but a separate C binding specification is also being produced <draft-ietf-cat-gaa-cbind-0n.txt>

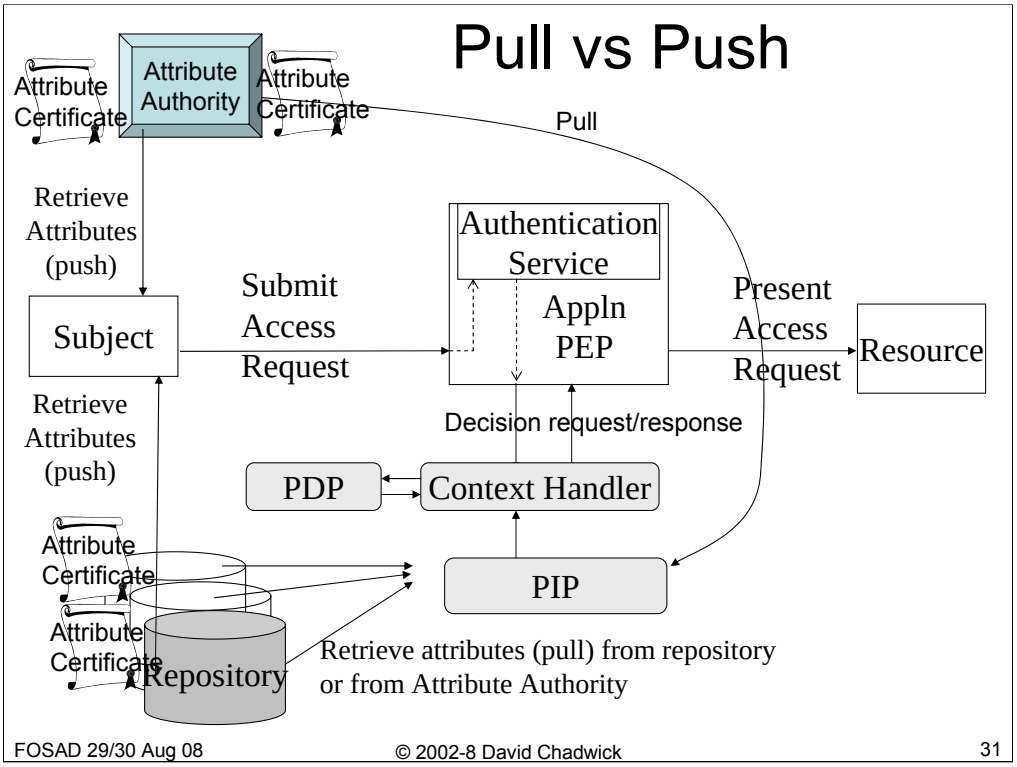
The PERMIS project has also defined its own API in Java, and is based on the OpenGroup's AZN API, but is a simplification of it, and is also specialised for use with X.509 Attribute Certificates.

The OASIS XACML specification has also defined an interface in XML, called the XACML request context and response context.



The slide shows the overall conceptual set of interactions, as described in XACMLv2 [1]. The PDP is initially loaded with the XACML policy, which it obtains from the PAP, prior to any user's requests being received (step 1). The user's access request is intercepted by the PEP (step 2), is authenticated, and any pushed credentials are validated and the attributes extracted (note that this is not within the scope of the XACML standard). The request and user attributes (in local format) are forwarded to the context handler (step 3), which may ask the PIP for additional attributes (steps 6 to 8) before passing the request to the PDP in standard format (step 4). If the PDP determines from the policy that additional attributes are still needed, it may ask the context handler for them (step 5). Optionally the context handler may also forward resource content (step 9) along with the additional attributes (step 10) to the PDP. The PDP makes a decision and returns it via the context handler (step 11) to the PEP (step 12). If the decision contains optional obligations they will be enforced by the obligations service (step 12).

[1] OASIS eXtensible Access Control Markup Language (XACML)" v2.0, 6 Dec 2004, available from http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml



The user's attributes can be pushed to the PDP by the user fetching them from either the Attribute Authority or a repository such as an LDAP directory, or the PDP can pull them itself from either place.

In the Web Services world, the Attribute Authority is sometimes referred to as the Security Token Service (STS) since this is the web service that creates digital credentials (or security tokens).

XACML Response

- XACML has 4 possible responses
- Permit – the request is granted
- Deny – the request is denied
- Not Applicable – the PDP does not have any policy that applies to this decision request
- Indeterminate – the PDP is unable to evaluate the requested access e.g. missing attributes, network errors while retrieving policies, syntax errors in the decision request or in the policy, etc.
- Not applicable and Indeterminate are types of deny, but the PEP may have an alternative PDP to ask in the case of Not Applicable, in which case it will simply ignore this response.

Deficiencies in XACML

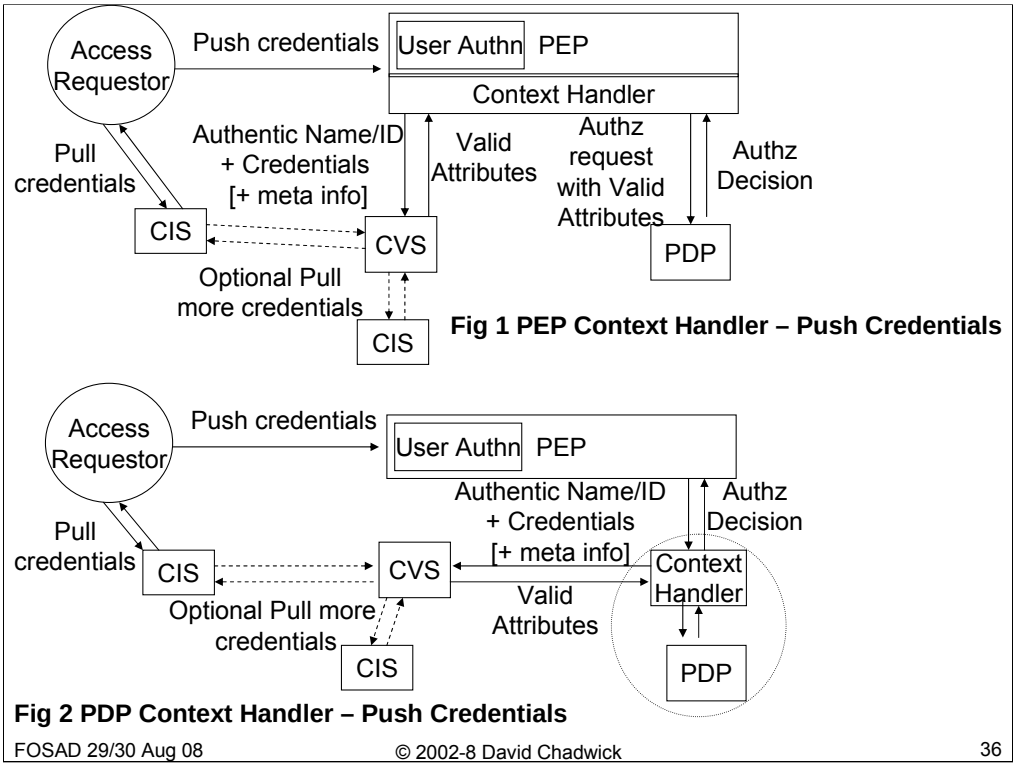
- XACML is only an access control policy language, it is not a standard for a full authz infrastructure/PMI e.g.
  - Does not define how user credentials are validated – only talks about user attributes
  - Does not say how policies are stored securely and retrieved
  - Does not cover auditing
  - Does not say how environmental attributes are securely obtained
- The XACMLv2 policy language also has some deficiencies e.g.
  - Does not specify what obligations are (other than anything you want them to be!)
  - Does not cover delegation of authority and policies for delegation of authority – this is in XACML v3

Grid Authz Infrastructure

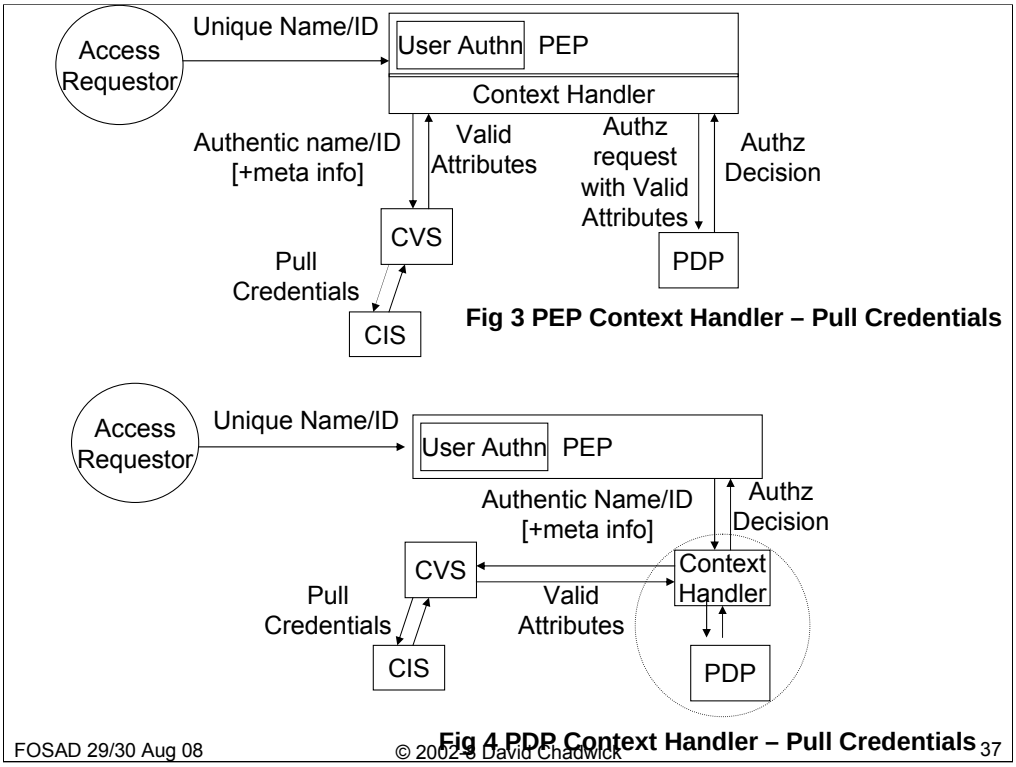
- The OGF has been working on an authz infrastructure for grid computing and has devised the following functional components and defined standard protocol profiles for how each of these components can communicate with the other ones
- CIS – Credential Issuing Service
- CVS – Credential Validation Service
- PDP – Policy Decision Point
- PEP – Policy Enforcement Point
- Context Handler – Marshalls arguments into correct format

In Grids we separate UA from PA

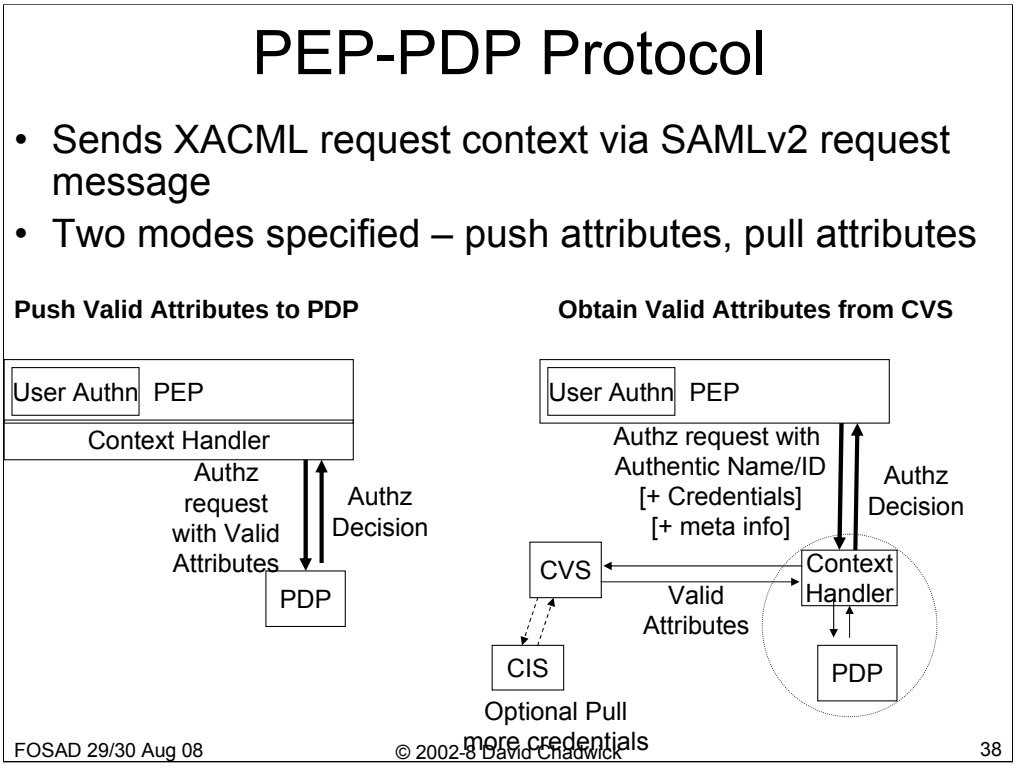
- In traditional RBAC systems both User-Attribute (UA) assignments and Permission-Attribute (PA) assignments are under the control of a single central authority
- In Grids and virtual organisations (VOs) we can no longer assume this is the case
- UA is performed by VO managers and other Attribute Authorities e.g. Government, Health Authorities etc.
- PA is performed by the resource owner (always) but some permissions may be delegated to VO managers
- This leads to the following model



These diagrams are taken from "**Functional Components of Grid Service Provider Authorisation Service Middleware**" April 2008, OGF Final Draft, ed D Chadwick.



Each of the above models have been implemented. The GridShib project has implemented Figures 1 and 3; whilst Globus Toolkit 4.1+ implements the PDP functionality of Figures 1 and 3. Figures 2 and 4 have been implmented by GTv3 onwards, as well as the PRIMEA PDP

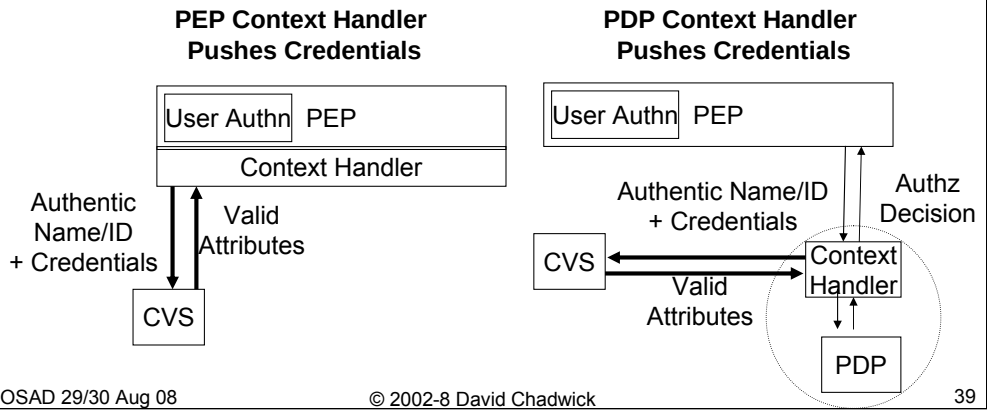


Examples. Fig 1. PERMIS in Push Mode

Fig 2.GGF OGSA Authz Protocol

CVS Push Protocol

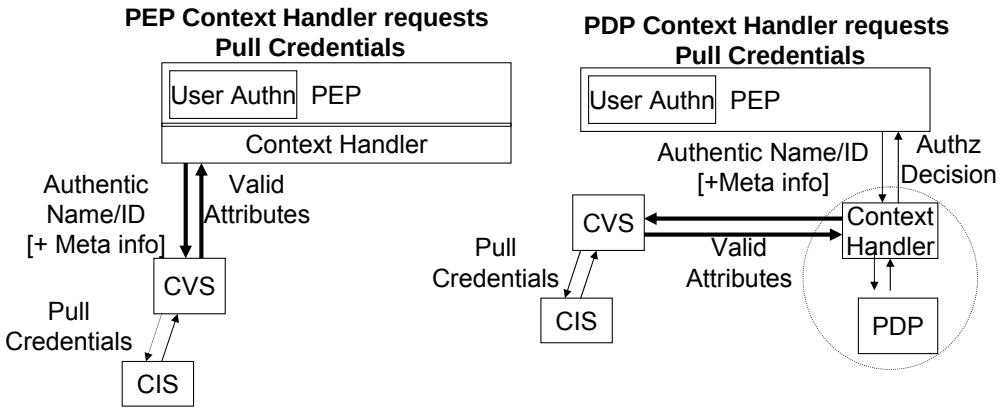
- SAML attribute assertions carried by WS-Trust and XACML attributes returned.
- Can work in three modes – push credentials or pull credentials or pull and push – using WS-Trust dialect
- Called by Context Handler embedded in PEP or PDP



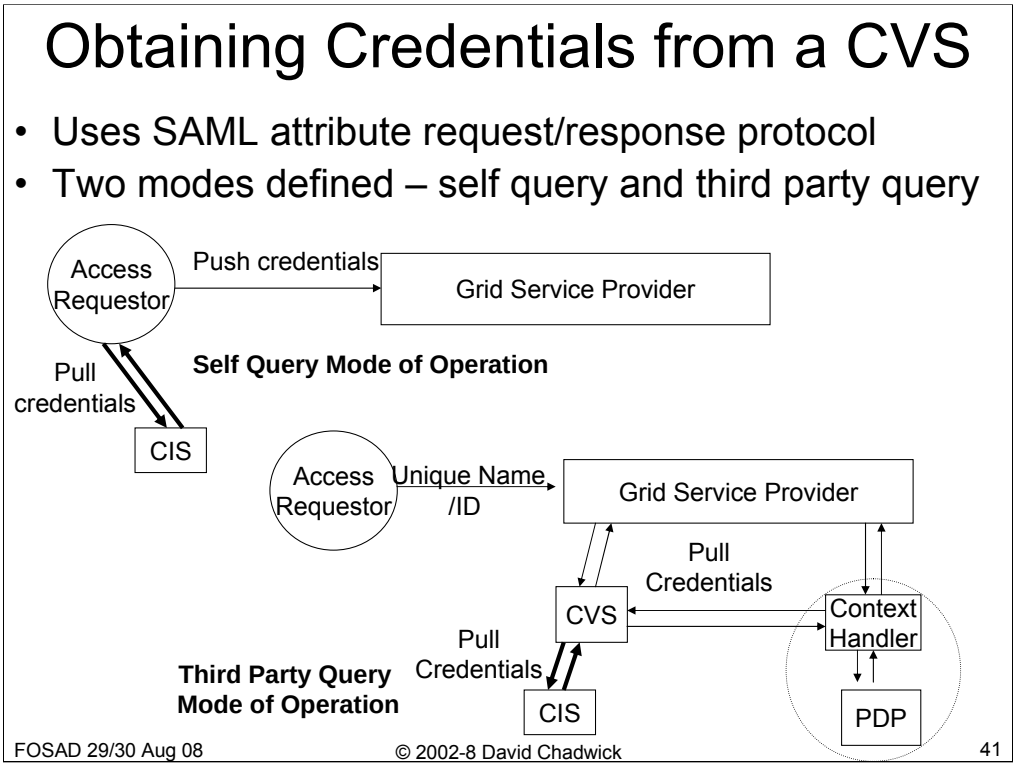
FOSAD 29/30 Aug 08 39

CVS Pulls Credentials

- SAML attribute assertion with no attributes carried by WS-Trust



FOSAD 29/30 Aug 08 © 2002-8 David Chadwick 40



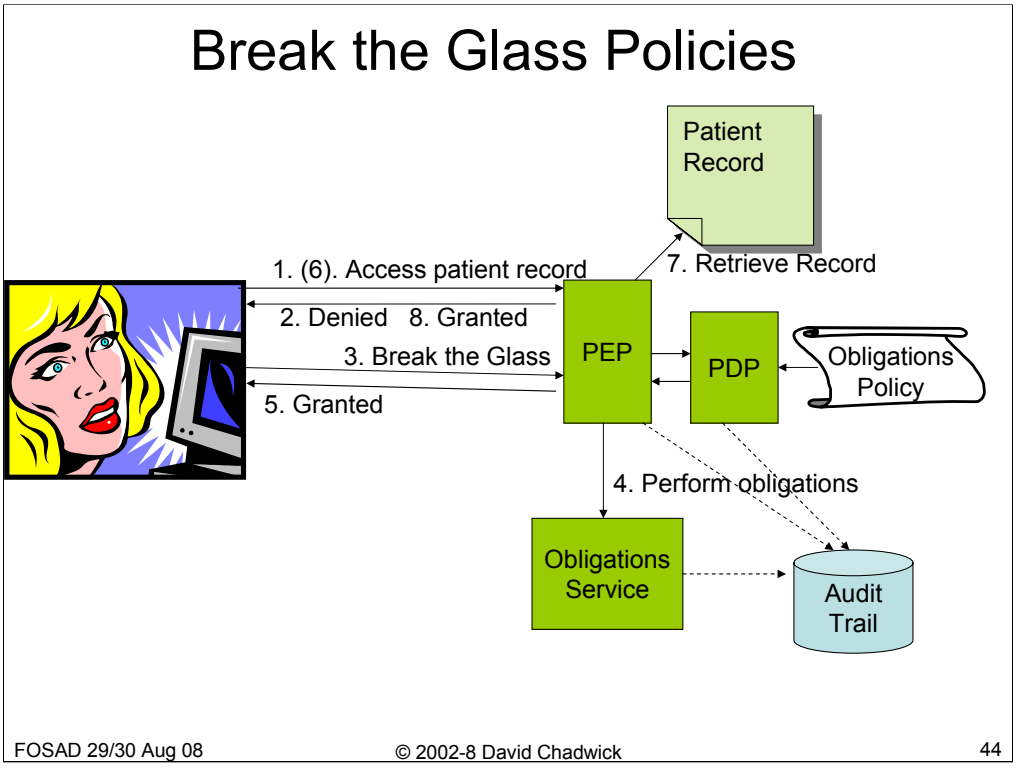
Currently there are privacy issues with the Third Party Mode, see next slide

Privacy Issues with 3rd Party Mode

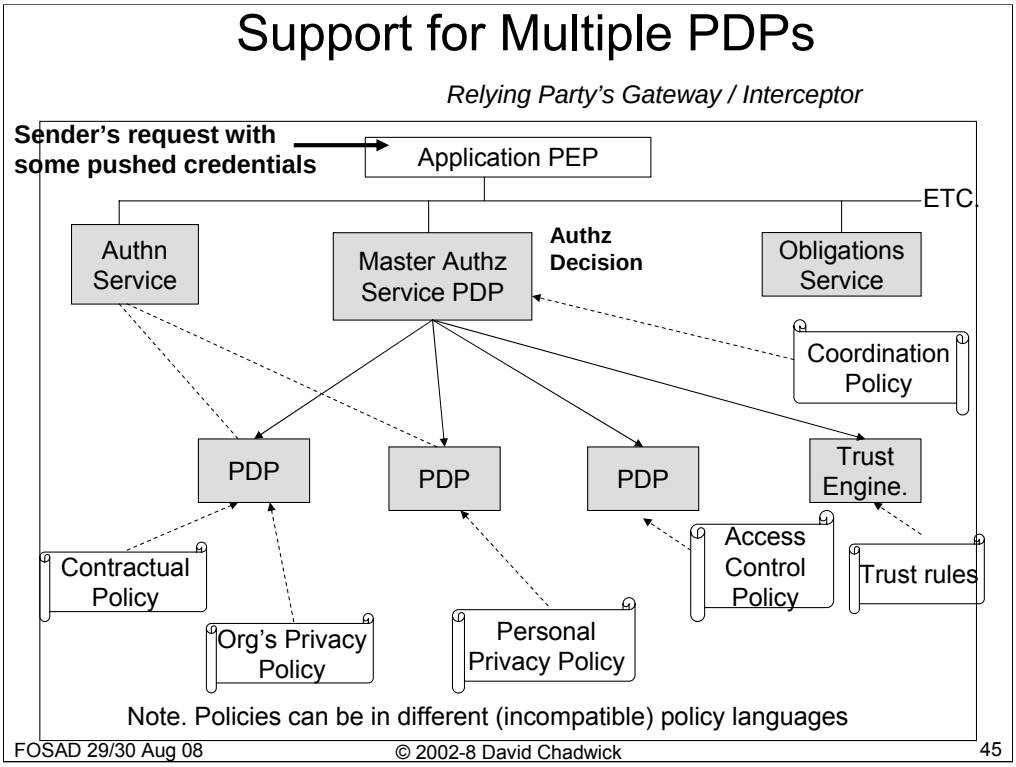
- How does CIS know user has given consent for his/her attributes to be retrieved?
  - Ans. Must have Attribute Release Policy at CIS i.e. a PDP controlling access to the CIS
- How does CIS know that user is currently accessing the Grid when the Grid pulls the user's credentials?
  - Current Answer is TRUST. The CIS must trust the Grid middleware/resource owner to act properly. But some say this is not sufficient. We need delegation of authority from the user to the Grid middleware for it to act on user's behalf at time of access. UNRESOLVED
- N.B. Neither are issues when user pulls credentials him/her self

What Research and Standardisation Work is Still To Do?

- Standardising the attributes that are passed
- Standardising the obligations that are returned
- Standardising a protocol for talking to an Obligations Service
- Break the Glass Policies
- Support for Multiple PDPs executing different policies from different authorities (e.g. user policy, resource owner policy, VO policy, government policy etc.) and the policy combining rules
- Coordinated Decision Making
- Retrieving attributes from multiple sources
- Level of Authentication/Assurance (LOA)
- Support for Sticky Policies
- Building application independent PEPs and obligation services



Break the Glass is both a resource (Boolean can be set to True from initial value of False) and a condition on access. In order to access the resource "Break the Glass" any conditions can be set in the Access Control Policy, including any LoA value and user roles/attributes. The granularity of the resource can be one BTG per user, one per resource, one per domain etc. We have fully described this in our paper at the Policy 2006 conference.



Since different policy languages e.g. EPAL, XACML, Ponder, PERMIS, P3P support different features e.g. obligations, delegation of authority, least privileges, separation of duties, etc then it is likely that different PDPs and policy languages will co-exist for years to come. Providing they all support the same request-response interface then it is possible to integrate them all using a Master PDP.

Coordinated Decision Making

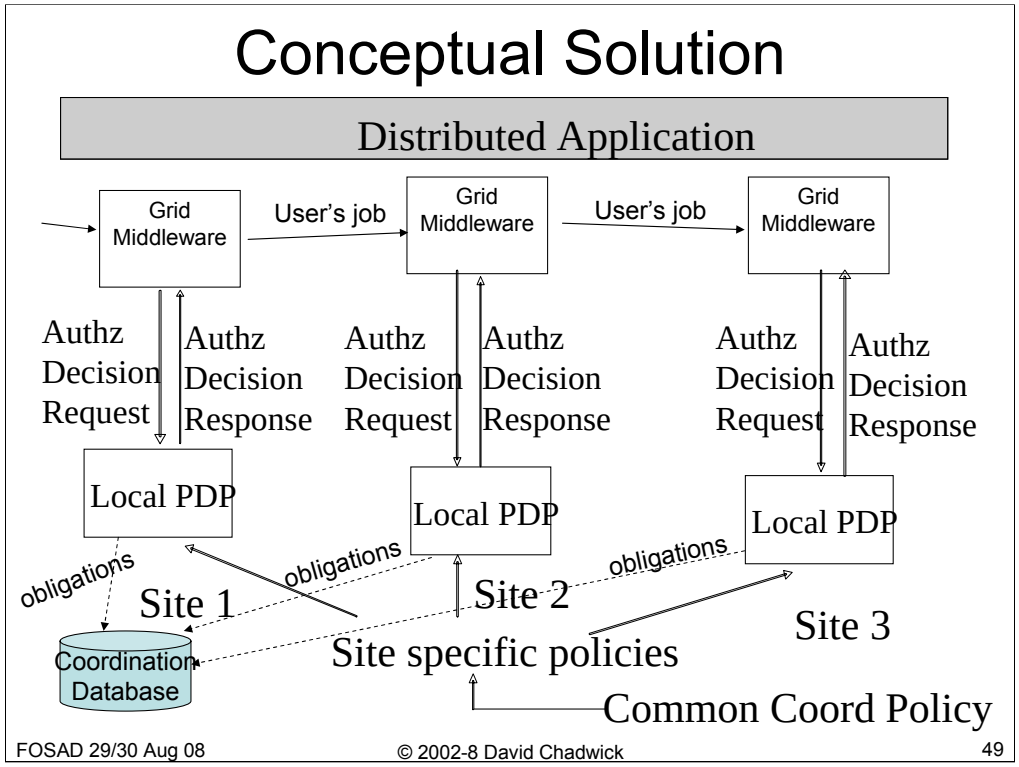
- Motivation/Problem Statement
- Sometimes one access control decision depends upon prior decisions
  - E.g. You can only draw £250 from ATM machines in a day
  - E.g. You are only entitled to use 5GB memory per grid job
- Decision may depend upon previous decisions at the same or different resources in the distributed system
- Relatively easy to solve if only one PDP is involved
  - Have a stateful PDP
  - Use existing PEP – PDP protocol from all nodes in Grid

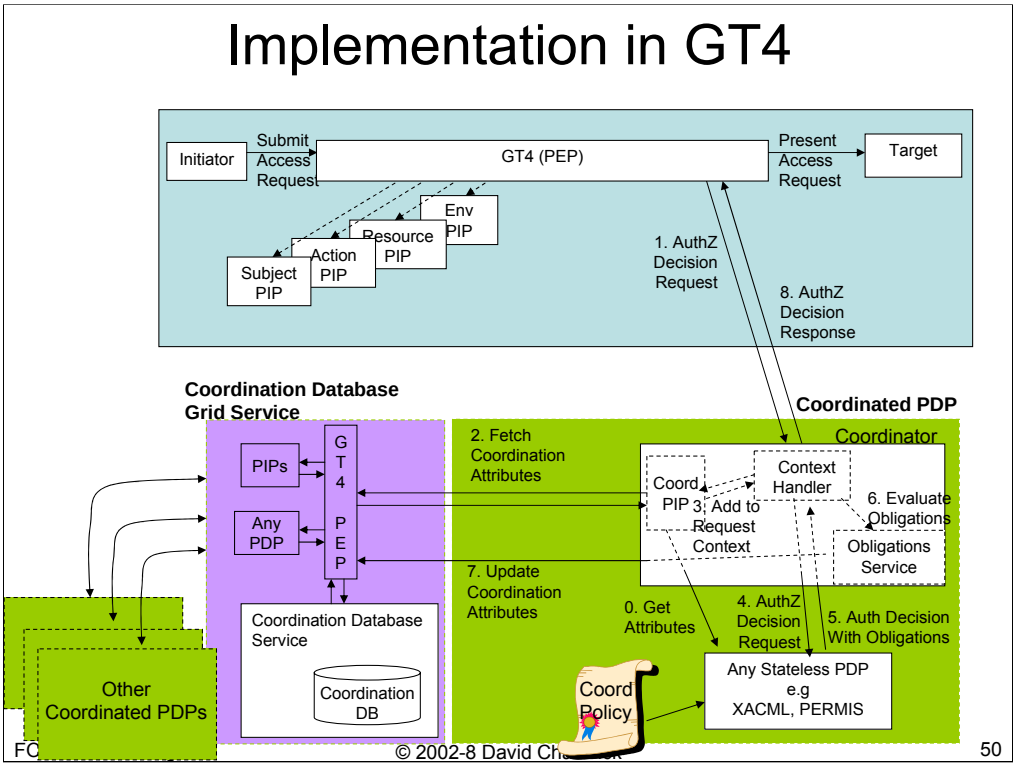
Disadvantages of one PDP Solution

- Requires a stateful PDP, but most PDPs (e.g. XACML) today are stateless
- In many VOs and grids there are multiple PDPs each with own policies
- Who would define policy for the entire Grid?
- Conclusion. We need to share state information between all the PDPs in the Grid, whilst using stateless PDPs! A dilemma

Conceptual Solution in Brief

- Store state information in *coordination* attributes of a *coordination* object
- Introduce a coordination policy for the distributed application (which each site can include as part of its access control policy)
  - Access control decisions will then depend upon values of these coordination attributes [as well as subject, resource, action and environmental attributes]
  - Obligations are used to update these coordination attributes
- Implement coordination object and attributes in a database grid service (DB provides stable storage, fast lookup, distribution, replication etc.)


Conceptual Solution
Distributed Application
Grid Middleware
User's job
Grid Middleware
User's job
Grid Middleware
Authz Decision Request
Authz Decision Response
Authz Decision Request
Authz Decision Response
Authz Decision Request
Authz Decision Response
Local PDP
Local PDP
Local PDP
obligations
obligations
obligations
Site 1
Site 2
Site 3
Coordination Database
Site specific policies
Common Coord Policy
FOSAD 29/30 Aug 08
© 2002-8 David Chadwick
49


Implementation in GT4
Initiator
Submit Access Request
GT4 (PEP)
Present Access Request
Target
Env PIP
Resource PIP
Action PIP
Subject PIP
1. AuthZ Decision Request
8. AuthZ Decision Response
Coordination Database Grid Service
Coordinated PDP
Coordinator
2. Fetch Coordination Attributes
PIPs
Any PDP
G T 4
P E P
Coord PIP
Context Handler
3. Add to Request Context
6. Evaluate Obligations
Obligations Service
7. Update Coordination Attributes
0. Get Attributes
4. AuthZ Decision Request
5. Auth Decision With Obligations
Coordination Database Service
Coordination DB
Other Coordinated PDPs
Coord Policy
Any Stateless PDP e.g XACML, PERMIS
© 2002-8 David Ch
50

Implementation Results

Time for 1 PDP to make a coordinated decision

Use of GSI	Response	Average time (ms)	STD DEV
With GSI	DENY	824	70
Without GSI	DENY	320	32
With GSI	GRANT	809	77
Without GSI	GRANT	345	34

Implementation Results

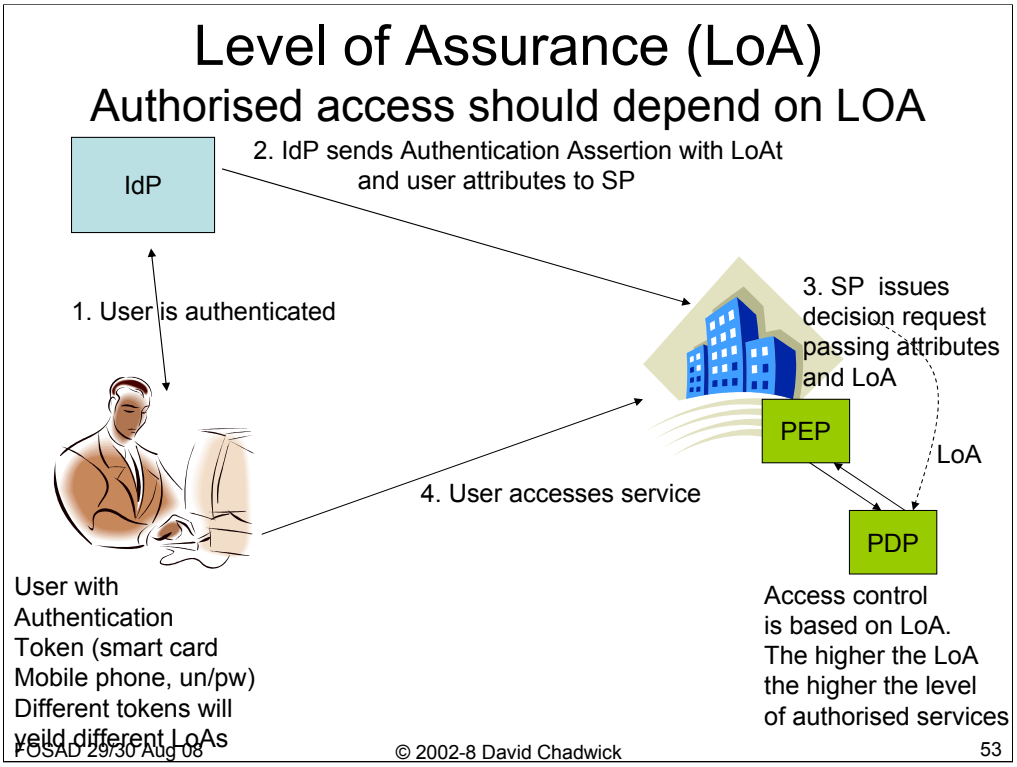
Time for multiple PDPs using GSI accessing the same coordination attribute

No of PDPs	PDP1	PDP2	PDP3	PDP4	PDP5
1	809±85	785±90	730±74	709±71	684±58
2	867±128	843±104			
3	1065±108	1052±121	1102±107		
4	1442±119	1414±111	1453±115	1456±119	
5	1790±149	1790±124	1812±130	1815±133	1819±124

FOSAD 29/30 Aug 08 52

A full description of coordinated decision making can be found in

David W Chadwick, Linying Su, Romain Laborde. “Coordinating Access Control in Grid Services”. Concurrency and Computation: Practice and Experience, Volume 20, Issue 9, Pages 1071-1094, 25 June 2008. Online version available from http://www3.interscience.wiley.com/cgi-bin/abstract/117347062/ABSTRACT

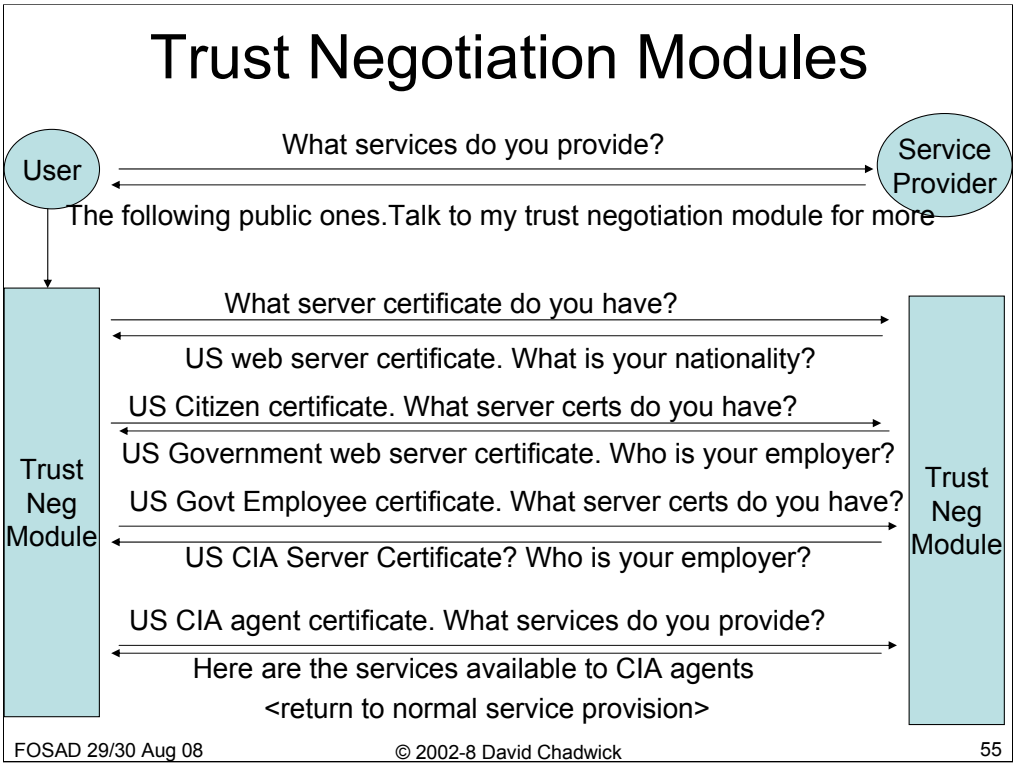


Note that the OASIS SAML group are working on an extension to SAML to carry the LOA attribute. So the protocol for carrying this should be standardised soon.

Using a digital certificate and PKI can yield LoAs from 1 to 4 depending upon the Certification Practice Statement of the CA.

Trust Negotiation

- When authorisation attributes are confidential, a user may not want to reveal them to a SP before knowing that the SP can be trusted.
- But the SP may also want to behave similarly and not reveal its attributes.
- E.g. A CIA agent in an Internet café in Moscow does not want to reveal his credentials to a CIA web server in the USA until he knows he is talking to the genuine web server and not a spoof one set up by the KGB
- Trust negotiation is the term used to refer to the gradual release of credentials by either party in order to mutually establish and build up trust between them


Trust Negotiation Modules
User
What services do you provide?
Service Provider
The following public ones.Talk to my trust negotiation module for more
Trust Neg Module
What server certificate do you have?
US web server certificate. What is your nationality?
US Citizen certificate. What server certs do you have?
US Government web server certificate. Who is your employer?
US Govt Employee certificate. What server certs do you have?
US CIA Server Certificate? Who is your employer?
US CIA agent certificate. What services do you provide?
Here are the services available to CIA agents
<return to normal service provision>
Trust Neg Module
FOSAD 29/30 Aug 08
© 2002-8 David Chadwick
55

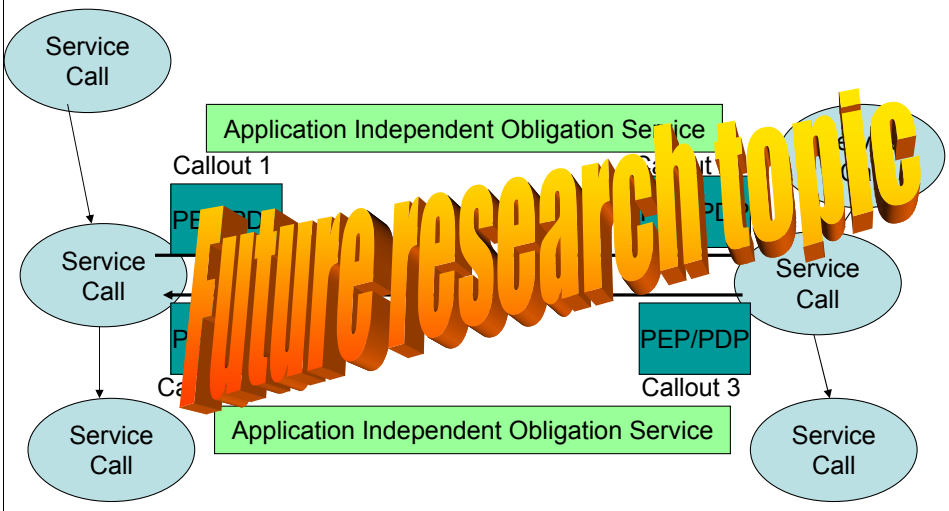
Sticky Policies

- Sometimes security policies must remain with the data they apply to and travel with it – so called sticky policies
- Examples:
  - a user has a privacy policy for his personal details (PII) and if the data holder transfers the user's PII to another business partner the user's privacy policy should go with it
  - a company document is confidential and should only be read by a restricted audience. When the document is transferred to one of the audience, the policy should remain with it to stop further transfers to unauthorised individuals

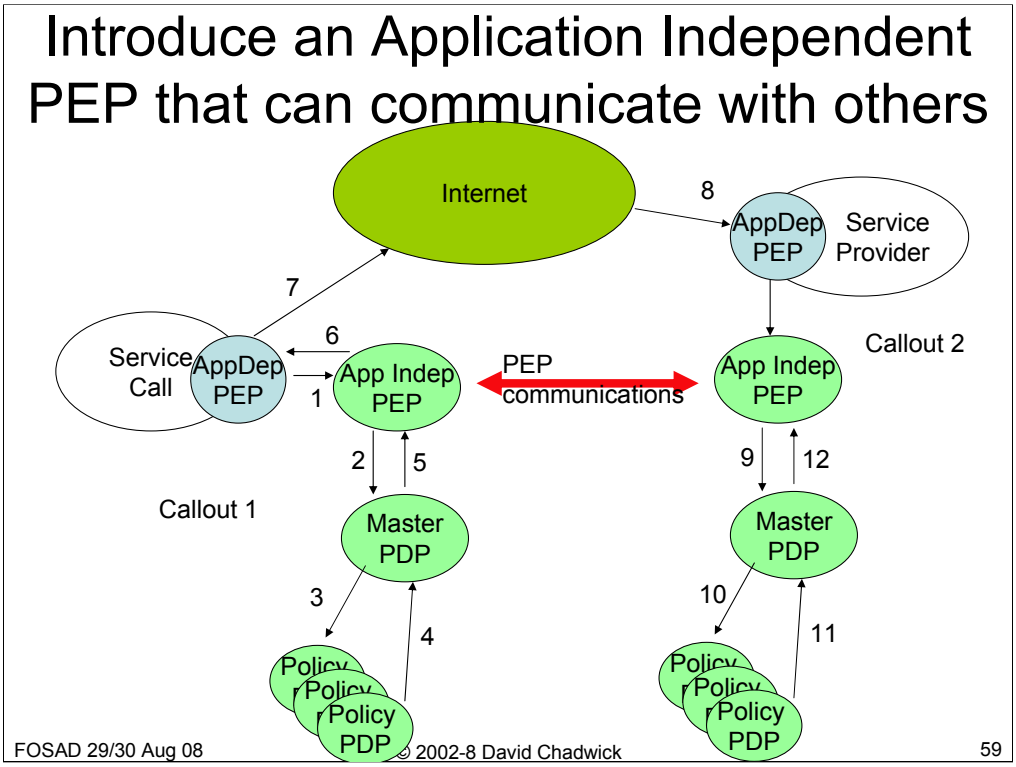
Removing Complexity From Application Developers

- All this complexity might be getting too much for application developers!!
- See if we can turn most of the security, privacy and trust into application independent middleware
- With just a simple call out by the application that returns granted or denied to the PEP each time
- → Introduce an application independent PEP
- This AIPEP can then be used to enforce sticky policies and obligations, as well as provide the trust negotiation capability in an application independent manner

Application Independent Obligations

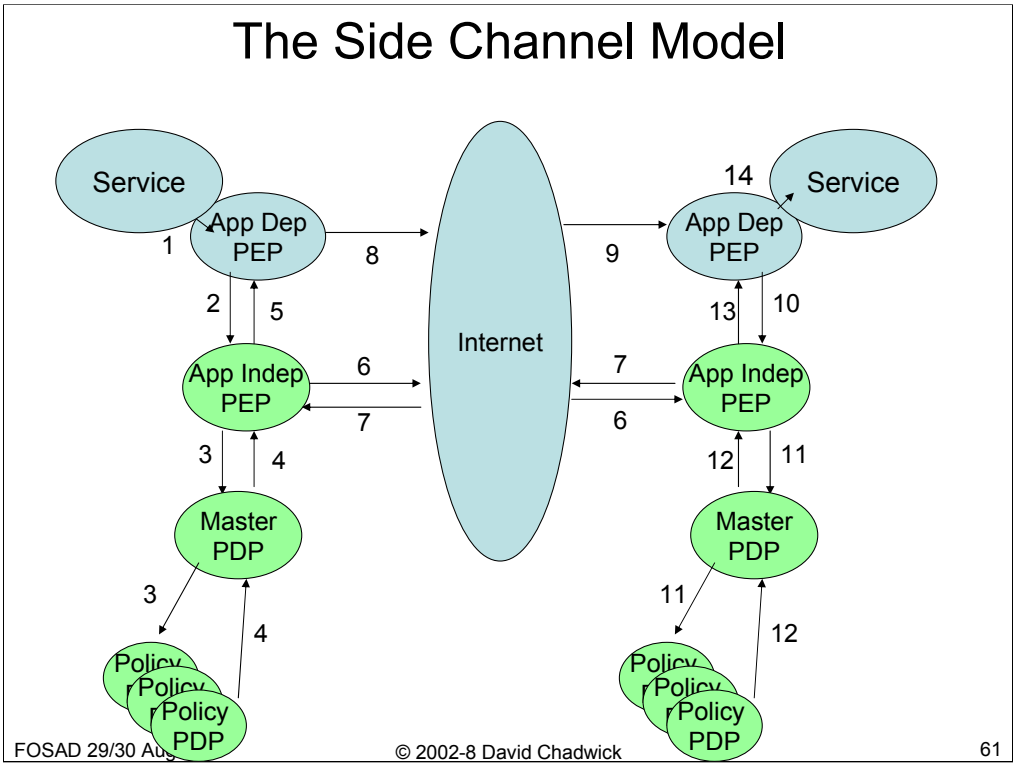


FOSAD 29/30 Aug 08 © 2002-8 David Chadwick 58

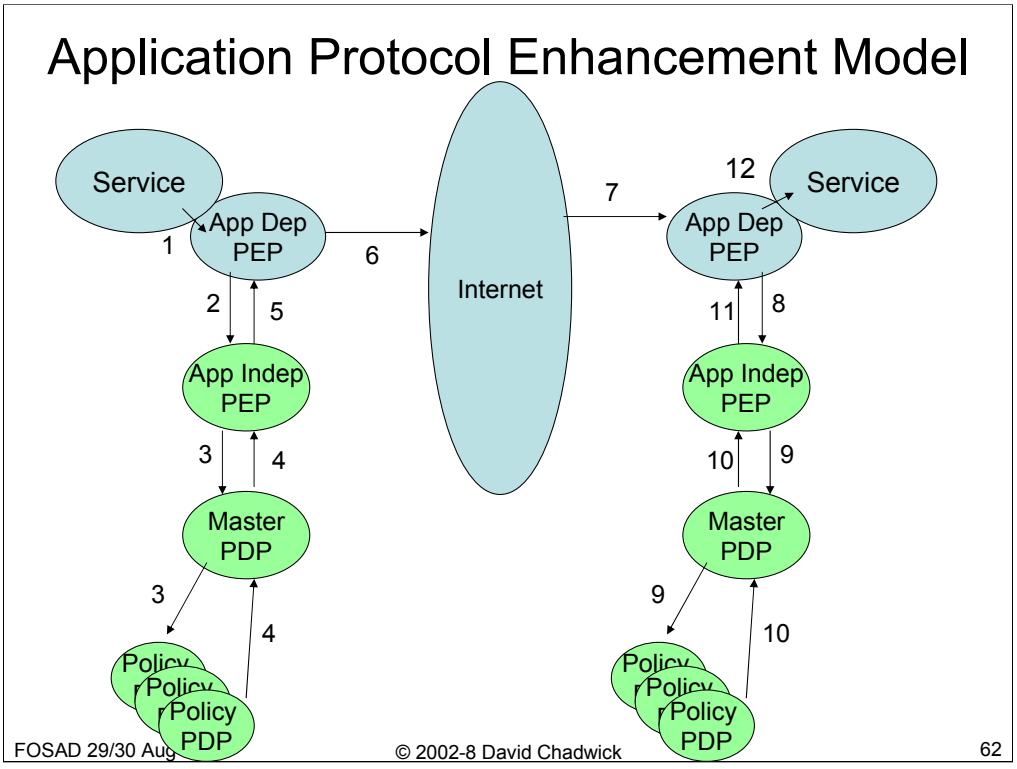

Introduce an Application Independent PEP that can communicate with others
Internet
8
AppDep PEP
Service Provider
7
6
Service Call
AppDep PEP
1
App Indep PEP
PEP communications
App Indep PEP
Callout 2
2
5
9
12
Callout 1
Master PDP
Master PDP
3
4
10
11
Policy PDP
Policy PDP
FOSAD 29/30 Aug 08
© 2002-8 David Chadwick
59

How should these AIPEPs communicate?

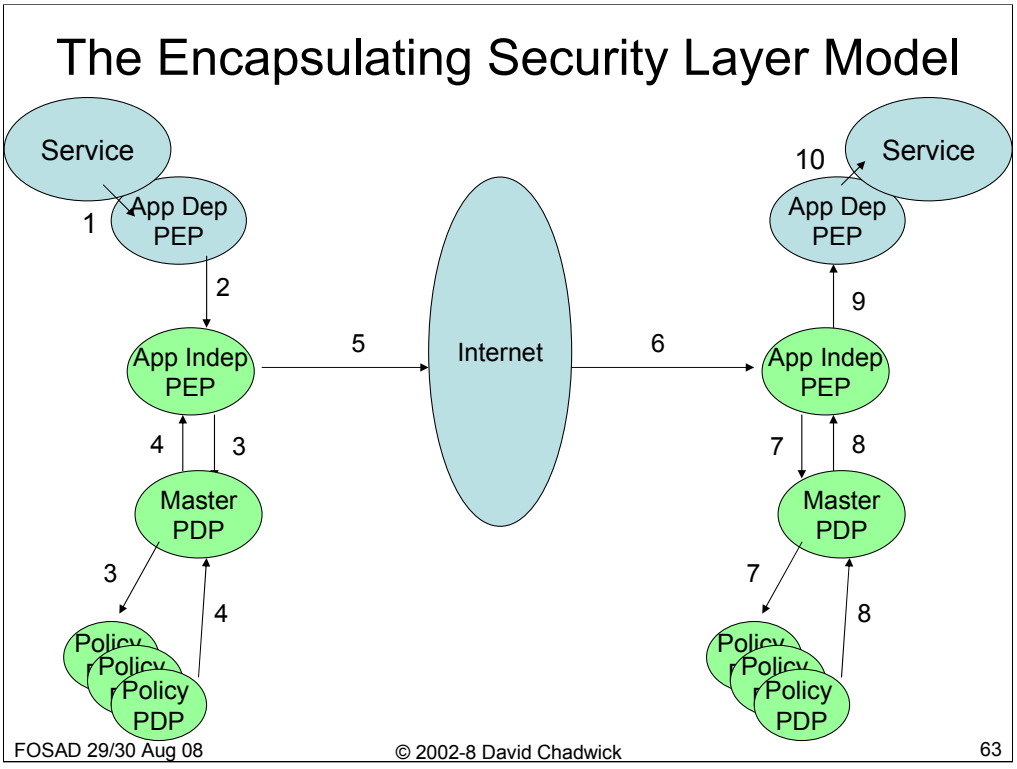
- There are several ways this could be achieved
- The AIPEPs talk directly to each other via a side channel – could be used for trust negotiation and sticky policy passing
- The AIPEPs pass policy related information back to the application for it to carry along with the application data
- The AIPEPs transfer the application data and the policy information on behalf of the application, and perform the trust negotiation



This is an extension of the previous research we did on coordinated decision making, where we have already demonstrated that our application independent authorisation infrastructure can communicate with other components of the same infrastructure.



An example of this strategy is S/MIME which has enhanced the MIME protocol to carry secure email messages. It is quite simple to integrate sticky policies into S/MIME messages as attributes that can be carried along with the messages. However it cannot perform trust negotiation



An example of this approach is SSL/TLS, which carries application layer data and protects it whilst in transfer. However, in this case the security layer will also carry authorisation policies and obligations between systems to ensure end to end enforcement of security policies. It can also perform the trust negotiation prior to establishing the service connection.

A Comparison of the 3 Models

Feature	Encapsulating security layer	Application protocol enhancement	Side channel
Requires modification to the application layer protocol	No	Yes, since extra policy information needs to be carried	No
Requires trust in the application dependent PEP	Yes, to parse the message into security blocks	Yes, to parse the message into security blocks and to carry the sticky policy between systems	Yes, to parse the message into security blocks
Requires changes to the application dependent PEP interface	Yes significantly, since the AIPEP will be transporting the application messages	Yes somewhat, since policies are being transferred across the interface	No
Has control over distributed system security	Yes complete control	Yes somewhat, but has to rely on application more	Yes, significant control
Implementation effort for application	Most complex	Easier	Easiest (no changes needed)
Implementation effort for application independent PEP	Most complex	Easiest	Medium difficulty
Flexible use of security in application data transfer between systems	Yes, can understand sticky policy and act accordingly	No, application must act independently of sticky policy	No, application must act independently of sticky policy
Supports Trust Negotiation	Yes	No	Yes

Conclusions

- Application authorisation infrastructures can be constructed in a modular fashion
- Several standards have already been specified for authz credentials and protocols for the different components to communicate
- Current active research and standardisation into more complex requirements such as sticky policies, multiple policies, break the glass policies, distributed policy enforcement etc using application independent components

- ANY LAST QUESTIONS?