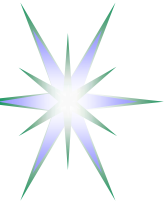
Authorisation Infrastructure for multidomain Network Resource Provisioning

Yuri Demchenko

System and Network Engineering Group
University of Amsterdam

OGSA-AUTHZ WG, OGF 23
2 June 2008, Barcelona



Outline

- AAA/AuthZ Architecture for Optical Network Resource Provisioning
- AAA/AuthZ mechanisms and functional components to support multidomain NRP
  - AuthZ ticket for extended AuthZ context management
  - Token Validation Service (TVS) and Token generation convention
- Reference Model for Obligations Handling (OHRM)
- XACML-NRP policy and attributes profile
- Using Identity Based Cryptography (IBC) for token key distribution at deployment stage

Background for this research

- EU funded Phosphorus Project “Lambda User Controlled Infrastructure for European Research” (EC Contract number 034115)
- University of Amsterdam SNE Group ongoing research on GAAA-AuthZ – Generic Authentication, Authorization, Accounting (GAAA) AuthZ Framework

(Optical) Network Resource Provisioning (NRP)

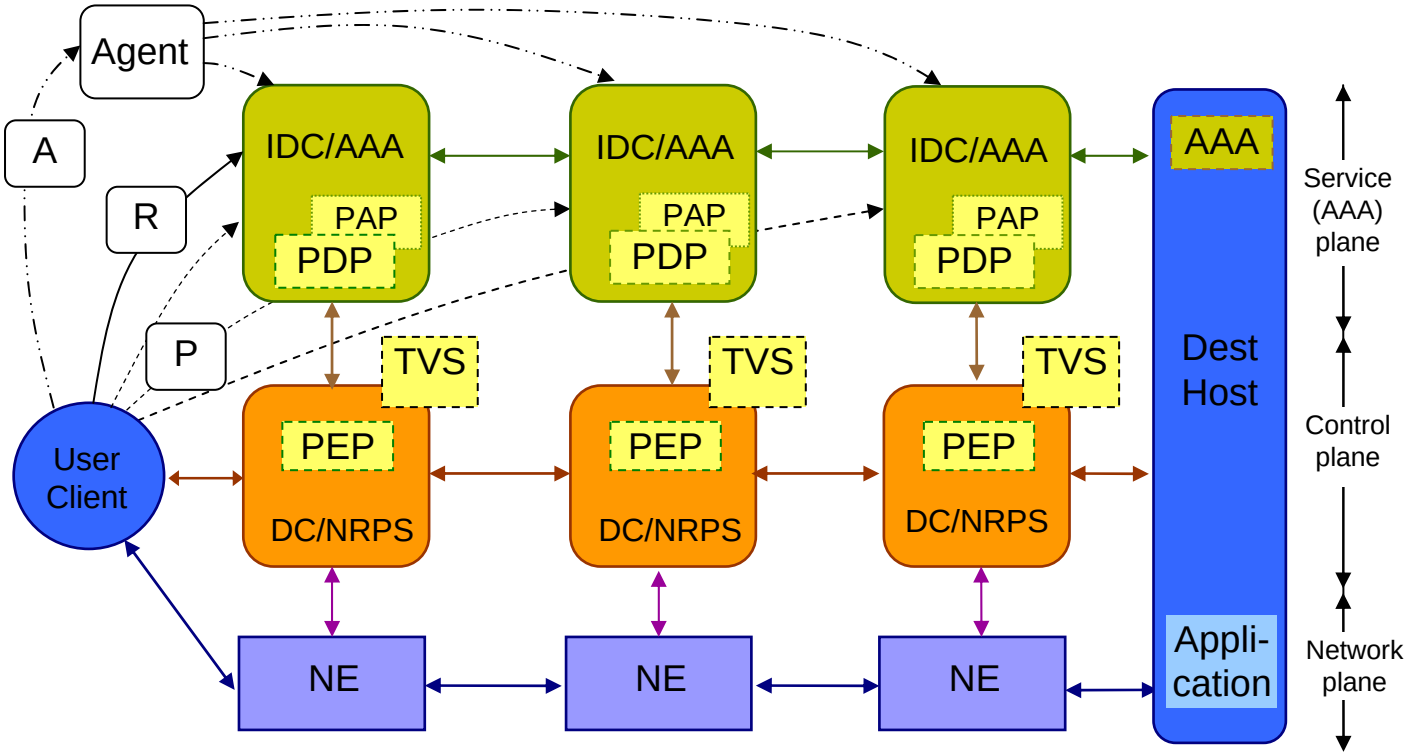
NRP as a use case of the general Complex Resource Provisioning (CRP)

- ONRP and Network on-demand provisioning
- Grid Computing Resource – Distributed and heterogeneous

3 stages/phases in NRP/CRP operation

- Reservation consisting of 3 basic steps
  - Resource Lookup
  - Resource composition (including options)
  - (Advance) Network resources reservation, including AuthZ/policy decision, and assigning a global reservation ID (GRI)
- Deployment (To be considered if it should be presented as a separate stage)
  - Confirmation – additional step that may be required to finalise reservation
- Access (to the reserved resource) or consumption (of the consumable resource)
  - Token or ticket based AuthZ decision enforcement

Multidomain Network/Complex Resource Provisioning



Provisioning sequences

- Agent (A)
- Polling (P)
- Relay (R)

Token based policy enforcement

GRI – Global Reservation ID

AuthZ tickets for multidomain context mngnt

IDC – Interdomain Controller

DC – Domain Controller

NRPS – Network Resource Provisioning System

AAA – AuthN, AuthZ, Accounting Server
PDP – Policy Decision Point
PEP – Policy Enforcement Point
TVS – Token Validation Service
KGS – Key Generation Service

AAA/AuthZ mechanisms and functional components to support multidomain NRP

The proposed AAA/security mechanisms and functional components to extend generic AAA AuthZ framework (PEP, PDP, PAP and operational sequences)

Token Validation Service (TVS) to enable token based policy enforcement

- Can be applied at all Networking layers (Service, Control and Data planes)
- *New proposed Pilot Token mechanism*

AuthZ ticket datamodel and format for extended AuthZ session management

- To allow extended AuthZ decision/session context communication between domains

Policy Obligation Handling Reference Model (OHRM)

- Used for account mapping, quota enforcement, accounting, etc.

XACML policy profile for NRP

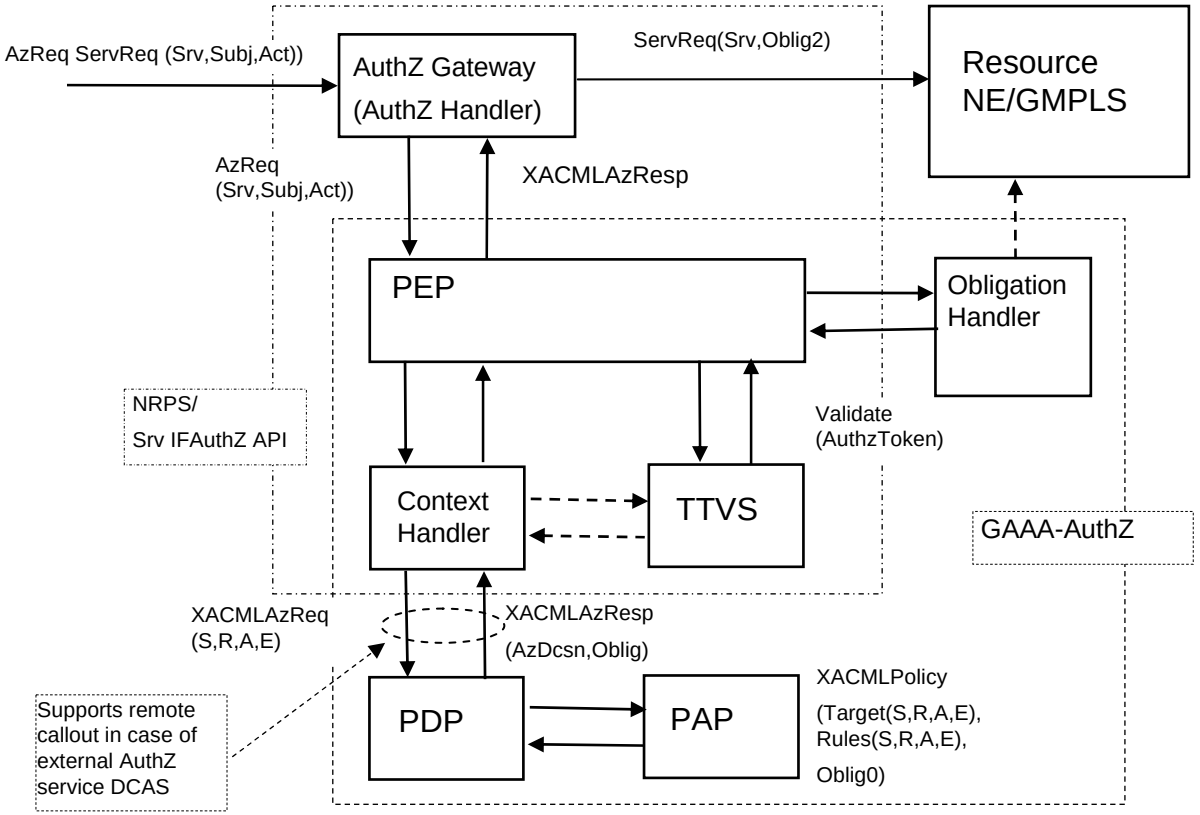
- Using reach functionality of the XACML policy format for complex network and Grid resources
- *Potentially may use path/topology information*

Identity Based Cryptography (IBC) use for token key distribution in inter-domain network resource provisioning will be investigated

The proposed architecture will allow smooth integration with other AuthZ frameworks as currently used and being developed by NREN and Grid community

- Can provide basic AAA/AuthZ functionality for each network layer DP, CP, SP

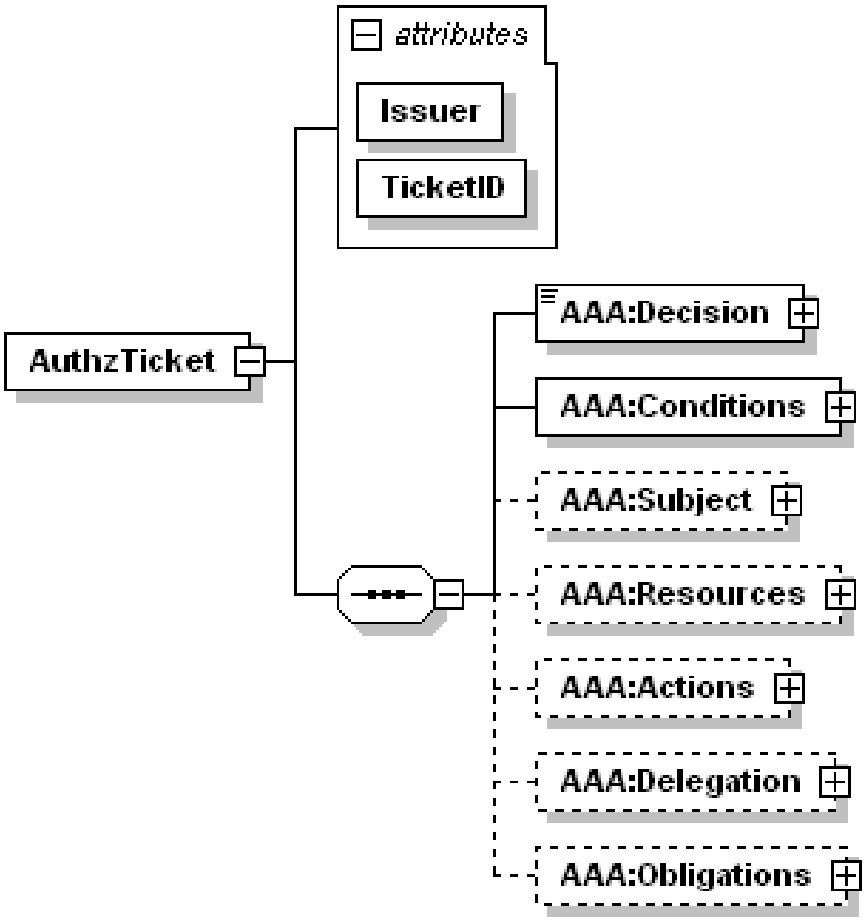
GAAA Toolkit pluggable AAA/AuthZ components



The proposed model intends to comply with both the generic AAA-AuthZ framework and XACML AuthZ model

ContextHandler functionality can be extended to support all communications between PEP-PDP and with other modules

AuthZ ticket/assertion for extended security context management – Data model (1) - Top elements



Required functionality to support multidomain provisioning scenarios

- Allows easy mapping to SAML and XACML related elements

Allows multiple Attributes format (semantics, namespaces)

Establish and maintain Trust relations between domains

- Including Delegation

Ensure Integrity of the AuthZ decision

- Keeps AuthN/AuthZ context
- Allow Obligated Decisions (e.g. XACML)

Confidentiality

- Creates a basis for user-controlled Secure session

AuthZ ticket main elements

<Decision> element - holds the PDP AuthZ decision bound to the requested resource or service expressed as the ResourceID attribute.

<Conditions> element - specifies the validity constrains for the ticket, including validity time and AuthZ session identification and additionally context

<ConditionAuthzSession> (extendable) - holds AuthZ session context

<Subject> complex element - contains all information related to the authenticated Subject who obtained permission to do the actions

<Role> - holds subject's capbilities

<SubjectConfirmationData> - typically holds AuthN context

<SubjectContext> (extendable) - provides additional security or session related information, e.g. Subject's VO, project, or federation.

<Resources>/<Resource> - contains resources list, access to which is granted by the ticket

<Actions>/<Action> complex element - contains actions which are permitted for the Subject or its delegates

<Delegation> element – defines who the permission and/or capability are delegated to: another DelegationSubjects or DelegationCommunity

- attributes define restriction on type and depth of delegation

<Obligations>/<Obligation> element - holds obligations that PEP/Resource should perform in conjunction with the current PDP decision.

AuthZ ticket format (proprietary) for extended security context management

```
<AAA:AuthzTicket xmlns:AAA="http://www.aaauthreach.org/ns/#AAA" Issuer="urn:cnl:trust:tickauth:pep"
    TicketID="cba06d1a9df148cf4200ef8f3e4fd2b3">
  <AAA:Decision ResourceID="http://resources.collaboratory.nl/Philips_XPS1">Permit</AAA:Decision>
    <!-- SAML mapping: <AuthorizationDecisionStatement Decision="*" Resource="*"> -->
  <AAA:Actions>
    <AAA:Action>cnl:actions:CtrlInstr</AAA:Action>     <!-- SAML mapping: <Action> -->
    <AAA:Action>cnl:actions:CtrlExper</AAA:Action>
  </AAA:Actions>
  <AAA:Subject Id="subject">
    <AAA:SubjectID>WHO740@users.collaboratory.nl</AAA:SubjectID>    <!-- SAML mapping: <Subject>/<NameIdentifier> -->
    <AAA:SubjectConfirmationData>IGhA11vwa8YQomTgB9Ege9JRNnld84AggaDkOb5WW4U=</AAA:SubjectConfirmationData>
     <!-- SAML mapping:  EXTENDED <SubjectConfirmationData/> -->
    <AAA:Role>analyst</AAA:Role>
     <!-- SAML mapping:  <Evidence>/<Assertion>/<AttributeStatement>/<Assertion>/<Attribute>/<AttributeValue> -->
    <AAA:SubjectContext>CNL2-XPS1-2005-02-02</AAA:SubjectContext>
     <!-- SAML mapping: <Evidence>/<Assertion>/<AttributeStatement>/<Assertion>/<Attribute>/<AttributeValue> -->
  </AAA:Subject>
  <AAA:Delegation MaxDelegationDepth="3" restriction="subjects">
     <!-- SAML mapping:  LIMITED <AudienceRestrictionCondition> (SAML1.1), or <ProxyRestriction>/<Audience> (SAML2.0)  -->
    <AAA:DelegationSubjects> <AAA:SubjectID>team-member-2</AAA:SubjectID> </AAA:DelegationSubjects>
  </AAA:Delegation>
  <AAA:Conditions NotBefore="2006-06-08T12:59:29.912Z" NotOnOrAfter="2006-06-09T12:59:29.912Z" renewal="no">
     <!-- SAML mapping: <Conditions NotBefore="*" NotOnOrAfter="*"> -->
    <AAA:ConditionAuthzSession PolicyRef="PolicyRef-GAAA-RBAC-test001" SessionID="JobXPS1-2006-001">
     <!-- SAML mapping: EXTENDED <SAMLConditionAuthzSession PolicyRef="*" SessionID="*"> -->
      <AAA:SessionData>put-session-data-Ctx-here</AAA:SessionData>  <!-- SAML EXTENDED: <SessionData/> -->
    </AAA:ConditionAuthzSession>
  </AAA:Conditions>
  <AAA:Obligations>
    <AAA:Obligation>put-policy-obligation(2)-here</AAA:Obligation>  <!-- SAML EXTENDED: <Advice>/<PolicyObligation> -->
    <AAA:Obligation>put-policy-obligation(1)-here</AAA:Obligation>
  </AAA:Obligations>
</AAA:AuthzTicket>
<ds:Signature> <ds:SignedInfo/> <ds:SignatureValue>e4E27kNwEXoVdnXIBpGVjpaBGVY71Nypos...</ds:SignatureValue></ds:Signature>
```

XML token format

```
<AAA:AuthzToken xmlns:AAA="http://www.aaauthreach.org/ns/#AAA"
   Issuer="urn:aaa:gaaapi:token:TVS"
   SessionId="a9bcf23e70dc0a0cd992bd24e37404c9e1709afb"
   TokenId="d1384ab54bd464d95549ee65cb172eb7">
<AAA:TokenValue>ebd93120d4337bc3b959b2053e25ca5271a1c17e</AAA:TokenValue>
    <AAA:Conditions NotBefore="2007-08-12T16:00:29.593Z" NotOnOrAfter="2007-
   08-13T16:00:29.593Z"/>
</AAA:AuthzToken>
```

where

SessionId = GRI (Global Reservation Id)

TokenId – unique identifier (serving for logging and accountability)

TokenValue – generated securely from GRI or AuthzTicket (digital SignatureValue)

- The element <TokenValue> and attributes SessionId and TokenId are mandatory, and the element <Conditions> and attributes Issuer, NotBefore, NotOnOrAfter are optional
- Binary token contains just two values – TokenValue and GRI

TVS functionality – Basic and Extended

- Basic TVS functionality is checking validity of a token received from the PEP or AuthZ gateway/service
  - Extended TVS functionality should allow token re-building when sending dataflow to or requesting service from the next domain
  - Additionally, TVS may be required to support token or token key distribution at the reservation stage or at the stage of the reserved resource deployment
- Token building (TB) function generates token as derivative from the GRI and token key (which can also be generated based on GRI)
  - Additionally, TB should allow generating token dynamically using token key and variable dataflow data, e.g. IP packets payload as in case of TBS-IP
- TVS implementation should support both in-band dataflow token-based signalling and control plane signalling using XML-based tokens
  - To allow in-band token-based signalling, token key and token should be of fixed length

Handling access tokens with TVS

Using token for access control - Benefits

- Separates reservation and access stages
- More flexible comparing to AuthN/ID based approach
- Allows for multilayer token based access control

Proposed token handling conventions

- GRI is generated in the first domain or by the Reservation service
- Token is generated in the last domain and populated back to the requester
- All domains store/cache the confirmed GRI and returned token
- At the access stage the token is included into the request message and compared/validated by TVS with the stored token in each domain

Planned extensions

- Flexible GRI generation models (adding prefixes and suffixes)
- IBC key distribution model

TVS Implementation (using shared secret)

TVS is implemented as a component and a profile of the GAAA Toolkit GAAAPI package

- Supports token based AuthZ enforcement mechanism and infrastructure
- TVS related classes are organised as a **org.aaaarch.gaaapi.tvs** package. All interfaces are supported by corresponding method of the TVS.java class
- Can be integrated into the target network provisioning systems and applications, in particular OSCARS and DRAGON

The token generation and handling model is based on the shared secret HMAC-SHA1 algorithm:

`TokenKey = HMAC(GRI, tb_secret)`

where GRI – global reservation identifier,

tb_secret – shared Token Builder secret.

A token is created in a similar way but using TokenKey as a HMAC secret:

`TokenValue = HMAC(GRI, TokenKey)`

This algorithm allows for chaining token generation and validation process

`GRI-TokenKey-TokenValue => LRI-l_TokenKey-l_Token`

XACML Obligations - Definition

Policy Obligation is one of the policy enforcement mechanisms

- ***Obligations*** are a set of operations that must be performed by the ***PEP*** in conjunction with an ***authorization decision*** [XACML2.0]

Obligations semantics is not defined in the XACML policy language but left to bilateral agreement between a PAP and the PEP

PEPs that conform with XACMLv2.0 are required to deny access unless they understand and can discharge all of the <Obligations> elements associated with the applicable policy

Element <Obligations> / <Obligation>

- The <Obligation> element SHALL contain an ***identifier*** (in the form of URI) for the obligation and a set of attributes that form arguments of the action defined by the obligation. The FulfillOn attribute SHALL indicate the effect for which this obligation must be fulfilled by the PEP

XACML Obligations – Implementation suggestions

Obligation = Apply (TargetAttribute, Operation (Variables)), or
Obligation = Apply (TargetAttribute, Operation (Variables), Chronicle)

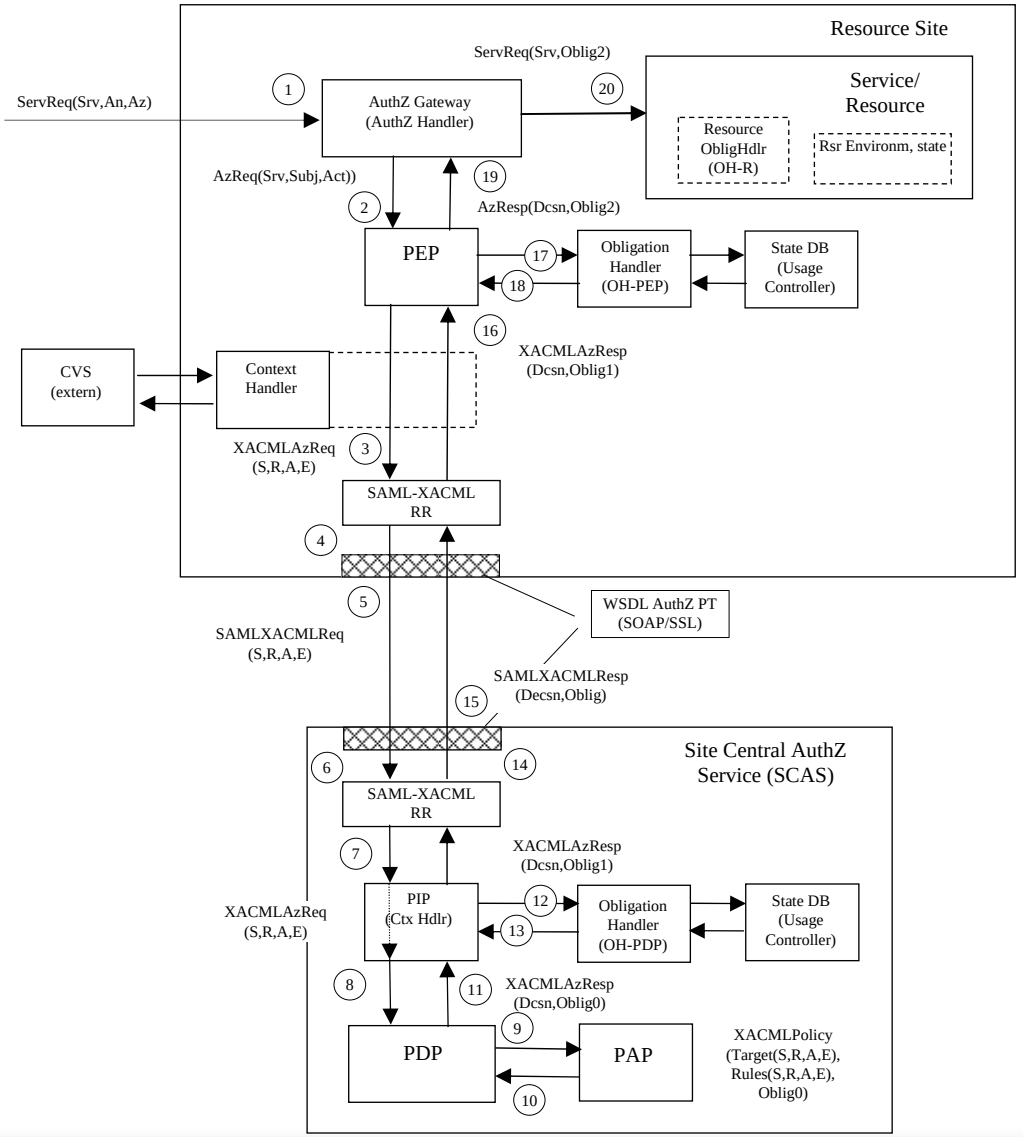
Obligations enforcement scenarios

- Obligations are enforced by PEP at the time of receiving obligated AuthZ decision from PDP
- Obligations are enforced at later time when the requestor accesses the resource or service
- Obligations are enforced before or after the resource or service accessed/delivered/consumed

Obligation handling model proposed in the process of interoperability workshop between OSG, EGEE, and Globus

- ObligationId (of type URI) has to be mapped to a specific handler that is called by the PEP
- Obligation parameter values are passed to handler
- Handler returns True/False that determines PEP's Permit/Deny

Proposed Obligations Handling Reference Model



Generic AuthZ service model

PEP – Policy Enforcement Point

PDP – Policy Decision Point

PAP – Policy Authority Point

OH – Obligation Handler

CtxHandler – Context Handler

(S, R, A, E) – components of the AuthZ request (Subject, Resource, Action, Environment)

Obligations Handling Stages

Obligation0 = tObligation => Obligation1 (“OK?”, (Attributes1 v Environments1))
=> Obligation2 (“OK?”, (Attributes2 v Environments2))
=> Obligation3 (Attributes3 v Environments3)

Obligation0 – (stateless or template)

Obligations are returned by the PDP in a form as they are written in the policy. These obligations can be also considered as a kind of templates or instructions, tObligation.

Obligation1 and Obligation 2

Obligations have been handled by Obligation handler at the SCAS/PDP side or at the PEP side, depending on implementation. Templates or instructions of the Obligation0 are replaced with the real attributes in Obligation1/2, e.g. in a form of “name-value” pair.

- The result of Obligations processing/enforcement is returned in a form of modified AuthzResponce (Obligation1) or global Resource environment changes
- Obligation handler should return notification about fulfilled obligated actions, e.g. in a form of Boolean value “False” or “True”, which will be taken into account by PEP or other processing module to finally permit or deny service request by PEP.
- Note. Obligation1 handling at the SCAS or PDP side allows stateful PDP/SCAS.

Obligation3

Final stage when an Obligation actually takes effect (Obligations “termination”). This is done by the Resource itself or by services managed/controlled by the Resource.

XACML Obligations – Examples of expression for pool account mapping in Grid – Option 1 (used in XACML-Grid)

```
<!-- Obligations format option 1 (simple): UID, GID explicitly mentioned as
   separate XML elements inside AttributeAssignment element -->
<xacml:Obligations>
   <xacml:Obligation
        ObligationId=http://authz-interop.org/xacml/obligation/uidgid
        FulfillOn="Permit">
     <xacml:AttributeAssignment
          AttributeId=http://authz-interop.org/xacml/attribute/posix-uid
          DataType="http://www.w3.org/2001/XMLSchema#integer">
       2501</xacml:AttributeAssignment>
     <xacml:AttributeAssignment
          AttributeId=http://authz-interop.org/xacml/attribute/posix-gid
          DataType="http://www.w3.org/2001/XMLSchema#integer">
       2101</xacml:AttributeAssignment>
   </xacml:Obligation>
<xacml:Obligations>
```

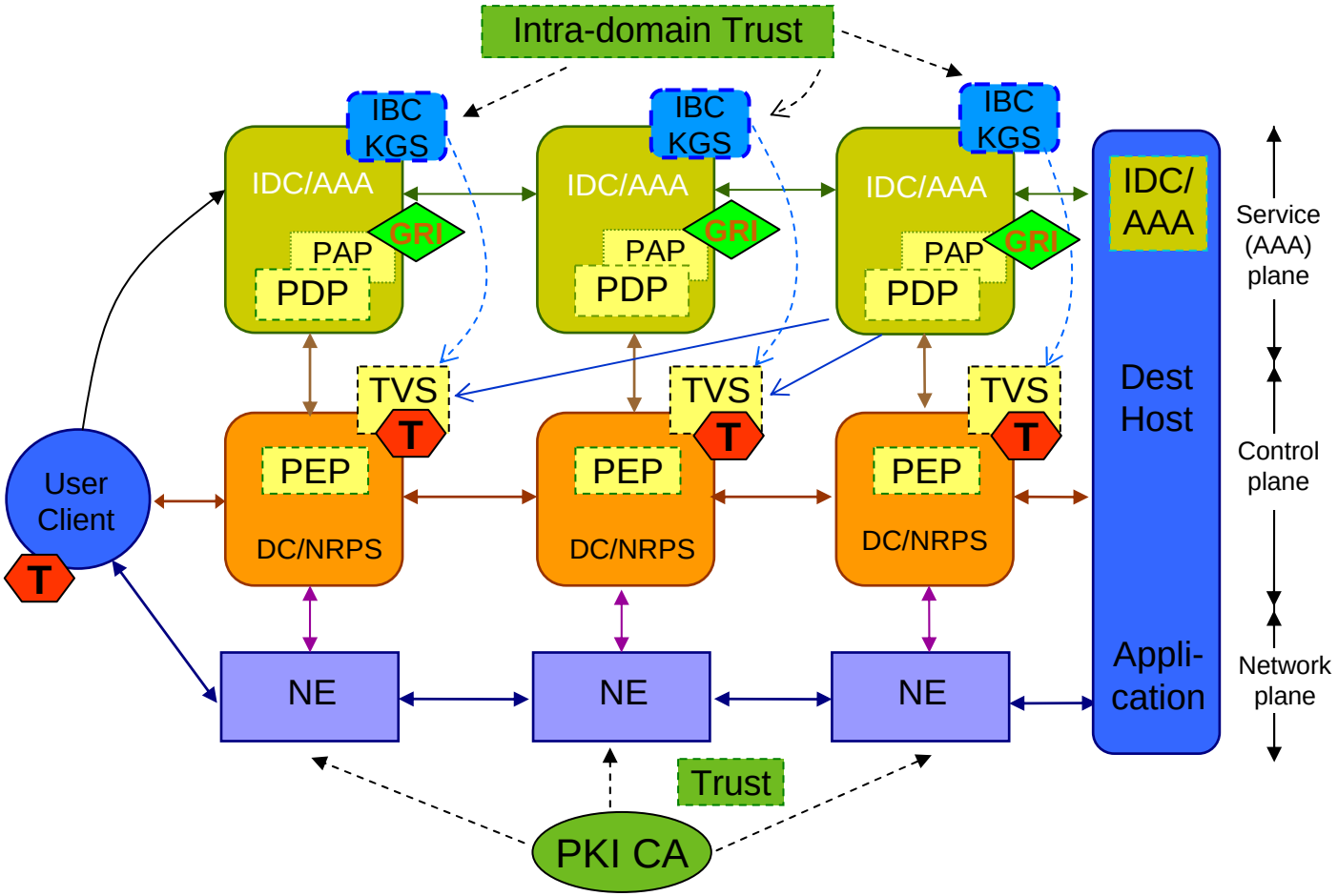
XACML Obligations – Examples of expression for pool account mapping in Grid – Option 2

```
<!-- Obligations format option 2: contains target attribute and what values to be assigned -->
<Obligations>
<Obligation ObligationId="http://authz-interop.org/xacml/obligation/map.poolaccount"
   FulfillOn="Permit">

<!-- This part specifies to what kind of attribute the next ‘map.to’ action is applied to -->
<AttributeAssignment
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute: requesting-subject"
DataType="http://www.w3.org/2001/XMLSchema#string">
    &lt;SubjectAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/&gt;
</AttributeAssignment>

<!-- This is actual account attribute name/value to which it should be mapped -->
<AttributeAssignment
    AttributeId="http://authz-interop.org/xacml/obligation/attribute/uidgid"
    DataType="http://www.w3.org/2001/XMLSchema#string">
  &lt;UnixId DataType="http://www.w3.org/2001/XMLSchema#string"&gt;
    okoeroo&gt;UnixId&gt;
  &lt; GroupPrimary DataType="http://www.w3.org/2001/XMLSchema#string"&gt;
    computergroup&gt;GroupPrimary&gt;
  &lt;GroupSecondary DataType="http://www.w3.org/2001/XMLSchema#string"&gt;
    datagroup&gt;GroupSecondary&gt;
</AttributeAssignment>
</Obligation>
</Obligations>
```

Identity Based Cryptography (IBC) infrastructure operation when distributing token keys in multidomain NRP



- Uses intra-domain trust relation without prior public key exchange
- Simplifies key management problem
- Allows flexibility in deploying/configuring intra-domain network path/infrastructure
- Used at deployment stage

- IBC KGS are setup independently but publish their public parameters

PKI vs Identity Based Cryptography (IBC)

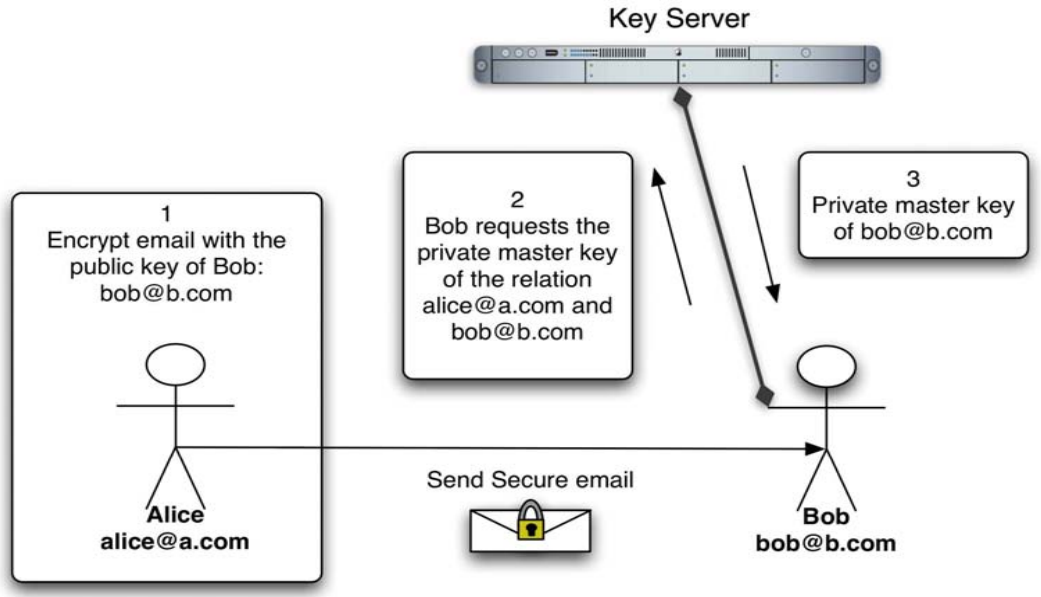
Uses publicly known remote entity's identity as a public key to send encrypted message or initiate security session

- Idea was proposed by Shamir in 1984 as an alternative to PKI and implementation by Dan Boneh and Matthew K. Franklin in 2001
- Identity can be email, domain name, IP address
- Allows conditional private key generation

Requires infrastructure different from PKI but domain based (doesn't require trusted 3rd party outside of domain)

- Parties may encrypt messages (or verify signatures) with no prior distribution of keys between individual participants
- Private key generation service (KGS)
  - Generates private key to registered/authenticated users/entities
  - To operate, the PKG first publishes a master public key, and retains the corresponding **master private key** (referred to as *master key*).
  - Given the master public key, any party can compute a public key corresponding to the identity *ID* by combining the master public key with the identity value.
- Exchange inter-domain trust management problem to intra-domain trust

Identity Based Cryptography (IBC) - Operation



Four algorithms form a complete IBE system (as proposed by Dan Boneh and Matthew K. Franklin):

Setup: This algorithm is run by the PKG one time for creating the whole IBE environment.

- The master key is kept secret and used to derive users' private keys, while the system parameters are made public. It accepts a security parameter k (i.e. binary length of key material) and outputs:
- A set P of system parameters, including the message space and ciphertext space M and C, a master key Km (master) .

Extract: This algorithm is run by the PKG when a user requests his private key.

- It takes as input P, Km and an identifier ID={0,1} and returns the private key D for user ID.
- Requires strong authentication and out of IBE model scope

Encrypt: Takes P, a message m={M} and ID={0,1} and outputs the encryption c={C}.

Decrypt: Accepts d, P and c={C} and returns m={M}

Future developments

- Defining XACML-NRP policy profile
  - For different Resource models or topology descriptions
    - Initial draft is available and will discussed at NML WG
  - Can be considered as extension to XACML-Grid profile
- Continue on AuthZ session management framework and AuthZ ticket format
- Implementation as GAAA-TK AuthZ library
  - First version will be released in July 2008 as Phosphorus project deliverable

Additional information

- XACML policy and Request-Response format
- Detailed AuthZ ticket data model

AAA AuthZ Request/Response messages format

Request (Subject (SubjectID, SubjectConfirmationData, SubjAttr, SubjCtx),
Resource (ResourceID), Action (ActionID))

where

SubjectID – Subject name in the form of simple name, URI or X.521

SubjectConfirmationData - AuthN token or Subject PKI Cert

SubjAttr – subject attributes e.g. roles or affiliation

SubjCtx - any additional information about Subject related to the Resource or Subject domain

Response (Result (Status, Obligations)):

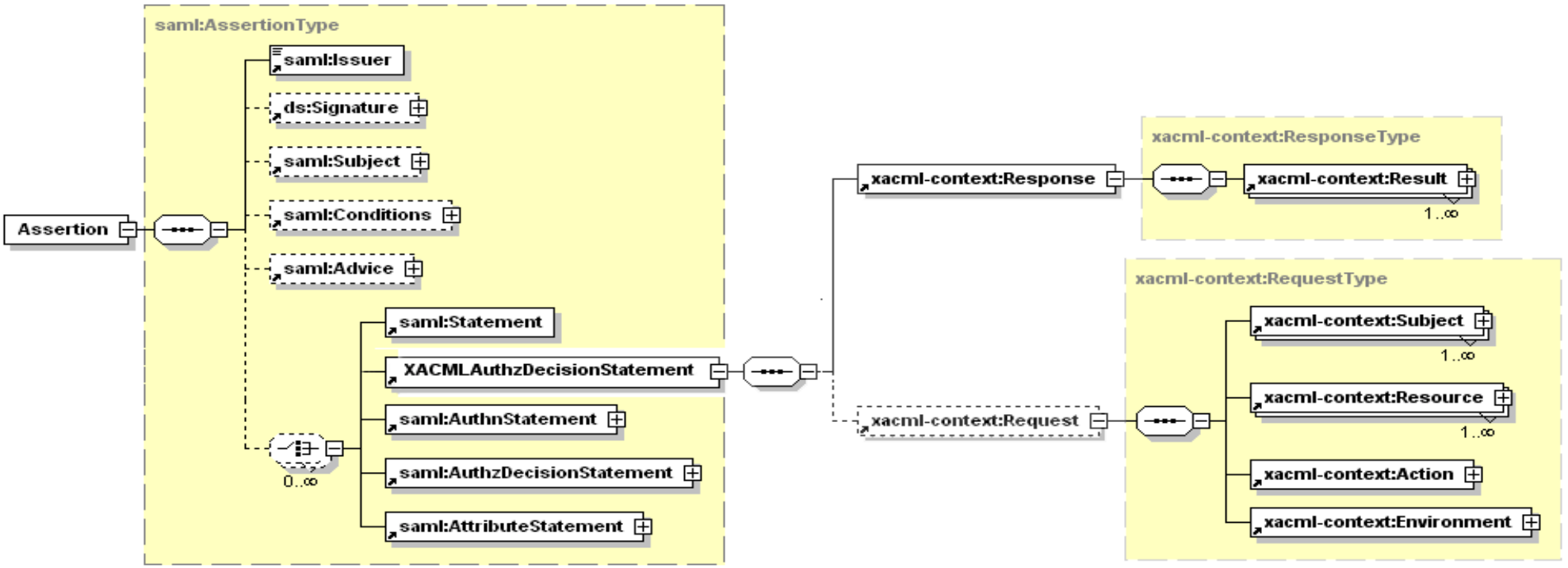
Suggested implementations

- XACML Request/Response messages, or
- SAML2.0 profile of XACML that encapsulates XACML Request/Response messages into SAML assertions and protocol
  - Recommended by OGF and GT-OSG-EGEE Interoperability Workshop

XACML Request message - Example

```
<xacml-context:Request xmlns:xacml="urn:oasis:names:tc:xacml:1.0:policy" xmlns:xacml-
    context="urn:oasis:names:tc:xacml:1.0:context" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context aaa-msg-xacml-01.xsd">
  <xacml-context:Subject Id="subject"    SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-
    subject">
    <xacml-context:Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
    DataType="http://www.w3.org/2001/XMLSchema#string" Issuer=" admin@gaaa.virtlab.nl ">
      <xacml-context:AttributeValue>WHO740@users.project.organisation.nl</xacml-context:AttributeValue>
    </xacml-context:Attribute>
    <xacml-context:Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subjconfdata"
    DataType="http://www.w3.org/2001/XMLSchema#string" Issuer=" admin@gaaa.virtlab.nl ">
      <xacml-context:AttributeValue>2SeDFGVHYTY83ZXxEdsweOP8Iok)yGHxVfHom90</xacml-context:AttributeValue>
    </xacml-context:Attribute>
    <xacml-context:Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:role"
    DataType="http://www.w3.org/2001/XMLSchema#string" Issuer=" admin@gaaa.virtlab.nl ">
      <xacml-context:AttributeValue>Analyst</xacml-context:AttributeValue>
    </xacml-context:Attribute>
  </xacml-context:Subject>
  <xacml-context:Resource>
    <xacml-context:Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="admin@gaaa.virtlab.nl">
      <xacml-context:AttributeValue>Resource-ID-here</xacml-context:AttributeValue>
    </xacml-context:Attribute>
  </xacml-context:Resource>
    <xacml-context:Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
    DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="admin@gaaa.collaboratory.nl">
      <xacml-context:AttributeValue>assign-time</xacml-context:AttributeValue>
    </xacml-context:Attribute>
  </xacml-context:Action>
</xacml-context:Request>
```

SAML-XACML Request/Response messages



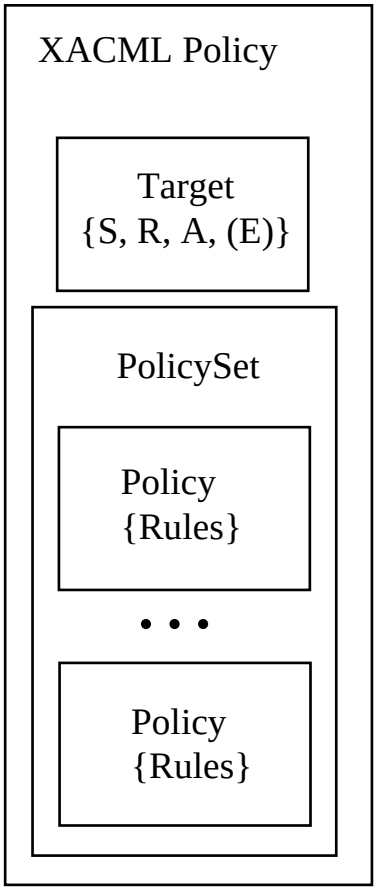
XACML Request-Response messages are enclosed into SAML2.0 Assertion SAML2.0 protocol messages

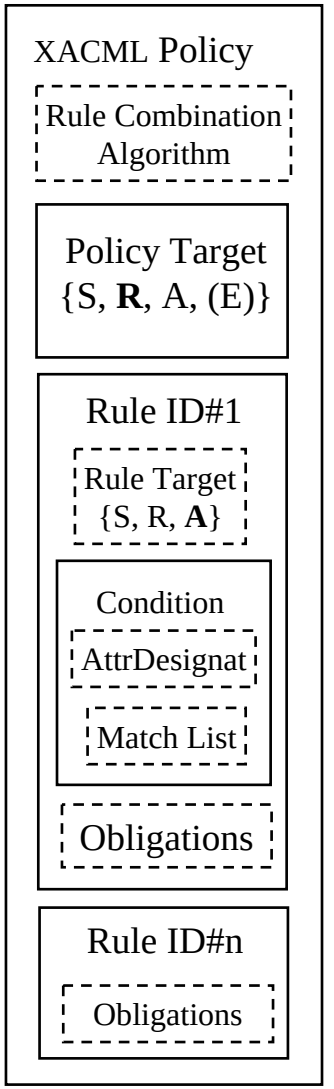
Extension library is available with OpenSAML2.0 and implemented in gLite (and Globus TK4.1?)

XACML Policy format

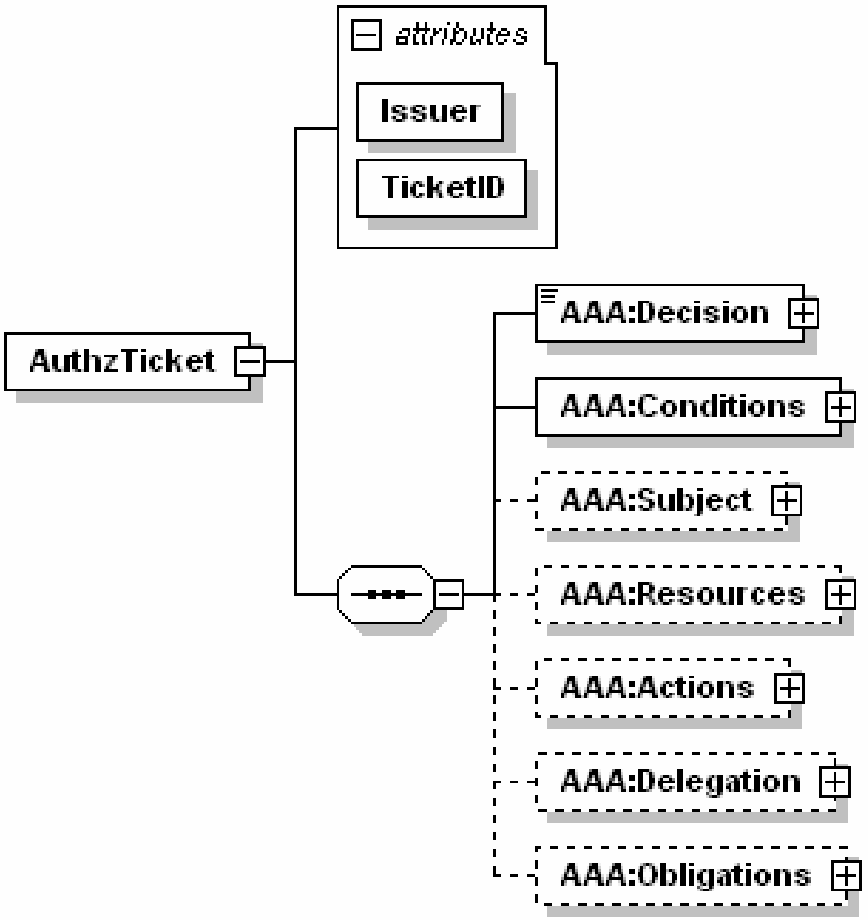
Policy consists of Policy Target and Rules

- Policy Target is defined for the tuple Subject-Resource-Action (-Environment)
- Policy Rule consists of Conditions and may contain Obligations





AuthZ ticket/assertion for extended security context management – Data model (1) - Top elements



Required functionality to support multidomain provisioning scenarios

- Allows easy mapping to SAML and XACML related elements

Allows multiple Attributes format (semantics, namespaces)

Establish and maintain Trust relations between domains

- Including Delegation

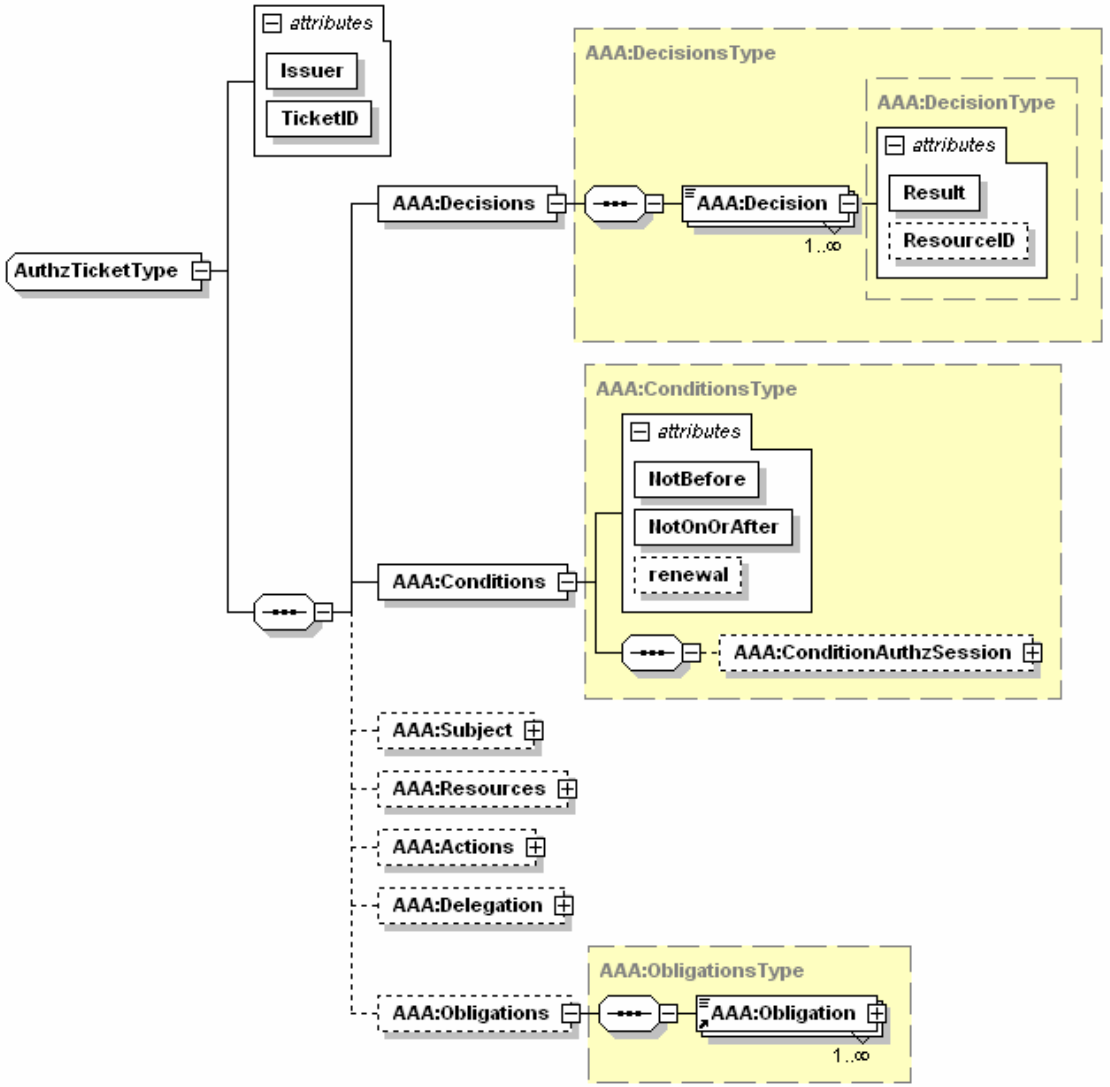
Ensure Integrity of the AuthZ decision

- Keeps AuthN/AuthZ context
- Allow Obligated Decisions (e.g. XACML)

Confidentiality

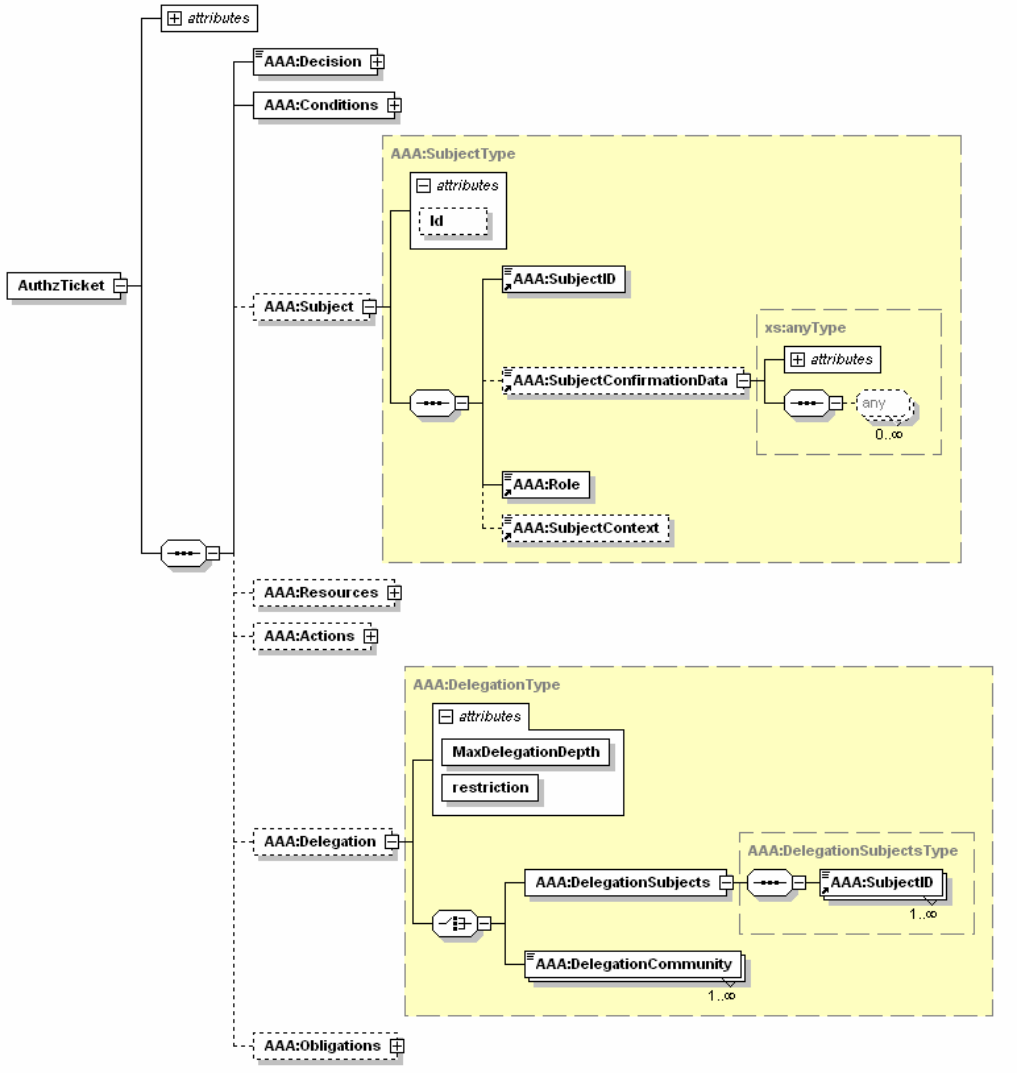
- Creates a basis for user-controlled Secure session

AuthZ ticket Data model (2) - Mandatory elements



- TicketID attribute
- Decisions element and ResourceID attribute
- Conditions Element and validity attributes
- Extensible element ConditionAuthzSession
  - Any AuthZ session related data

AuthZ ticket Data model (3) – Subject and Delegation elements



- Subject element to keep AuthN security context and Subject Attributes
- Delegation element to allow permissions/AuthZ decision delegation to other Subjects or groups/community

AuthZ ticket main elements

<Decision> element - holds the PDP AuthZ decision bound to the requested resource or service expressed as the ResourceID attribute.

<Conditions> element - specifies the validity constrains for the ticket, including validity time and AuthZ session identification and additionally context

<ConditionAuthzSession> (extendable) - holds AuthZ session context

<Subject> complex element - contains all information related to the authenticated Subject who obtained permission to do the actions

<Role> - holds subject's capbilities

<SubjectConfirmationData> - typically holds AuthN context

<SubjectContext> (extendable) - provides additional security or session related information, e.g. Subject's VO, project, or federation.

<Resources>/<Resource> - contains resources list, access to which is granted by the ticket

<Actions>/<Action> complex element - contains actions which are permitted for the Subject or its delegates

<Delegation> element – defines who the permission and/or capability are delegated to: another DelegationSubjects or DelegationCommunity

- attributes define restriction on type and depth of delegation

<Obligations>/<Obligation> element - holds obligations that PEP/Resource should perform in conjunction with the current PDP decision.

AuthZ ticket format (proprietary) for extended security context management

```
<AAA:AuthzTicket xmlns:AAA="http://www.aaauthreach.org/ns/#AAA" Issuer="urn:cnl:trust:tickauth:pep"
    TicketID="cba06d1a9df148cf4200ef8f3e4fd2b3">
  <AAA:Decision ResourceID="http://resources.collaboratory.nl/Philips_XPS1">Permit</AAA:Decision>
    <!-- SAML mapping: <AuthorizationDecisionStatement Decision="*" Resource="*"> -->
  <AAA:Actions>
    <AAA:Action>cnl:actions:CtrlInstr</AAA:Action>     <!-- SAML mapping: <Action> -->
    <AAA:Action>cnl:actions:CtrlExper</AAA:Action>
  </AAA:Actions>
  <AAA:Subject Id="subject">
    <AAA:SubjectID>WHO740@users.collaboratory.nl</AAA:SubjectID>    <!-- SAML mapping: <Subject>/<NameIdentifier> -->
    <AAA:SubjectConfirmationData>IGhA11vwa8YQomTgB9Ege9JRNnld84AggaDkOb5WW4U=</AAA:SubjectConfirmationData>
     <!-- SAML mapping:  EXTENDED <SubjectConfirmationData/> -->
    <AAA:Role>analyst</AAA:Role>
     <!-- SAML mapping:  <Evidence>/<Assertion>/<AttributeStatement>/<Assertion>/<Attribute>/<AttributeValue> -->
    <AAA:SubjectContext>CNL2-XPS1-2005-02-02</AAA:SubjectContext>
     <!-- SAML mapping: <Evidence>/<Assertion>/<AttributeStatement>/<Assertion>/<Attribute>/<AttributeValue> -->
  </AAA:Subject>
  <AAA:Delegation MaxDelegationDepth="3" restriction="subjects">
     <!-- SAML mapping:  LIMITED <AudienceRestrictionCondition> (SAML1.1), or <ProxyRestriction>/<Audience> (SAML2.0)  -->
    <AAA:DelegationSubjects> <AAA:SubjectID>team-member-2</AAA:SubjectID> </AAA:DelegationSubjects>
  </AAA:Delegation>
  <AAA:Conditions NotBefore="2006-06-08T12:59:29.912Z" NotOnOrAfter="2006-06-09T12:59:29.912Z" renewal="no">
     <!-- SAML mapping: <Conditions NotBefore="*" NotOnOrAfter="*"> -->
    <AAA:ConditionAuthzSession PolicyRef="PolicyRef-GAAA-RBAC-test001" SessionID="JobXPS1-2006-001">
     <!-- SAML mapping: EXTENDED <SAMLConditionAuthzSession PolicyRef="*" SessionID="*"> -->
      <AAA:SessionData>put-session-data-Ctx-here</AAA:SessionData>  <!-- SAML EXTENDED: <SessionData/> -->
    </AAA:ConditionAuthzSession>
  </AAA:Conditions>
  <AAA:Obligations>
    <AAA:Obligation>put-policy-obligation(2)-here</AAA:Obligation>  <!-- SAML EXTENDED: <Advice>/<PolicyObligation> -->
    <AAA:Obligation>put-policy-obligation(1)-here</AAA:Obligation>
  </AAA:Obligations>
</AAA:AuthzTicket>
<ds:Signature> <ds:SignedInfo/> <ds:SignatureValue>e4E27kNwEXoVdnXIBpGVjpaBGVY71Nypos...</ds:SignatureValue></ds:Signature>
```

TVS functional requirements

- Basic TVS functionality is checking validity of a token received from the PEP or AuthZ gateway/service
- TVS should allow easy integration into the control or data plane using simple API
- Extended TVS functionality should allow token re-building when sending dataflow to or requesting service from the next domain
- Additionally, TVS may be required to support token or token key distribution at the reservation stage or at the stage of the reserved resource deployment
- Token building (TB) function should allow generating token key and token as derivative from the GRI
  - Additionally, TB should allow generating token dynamically using token key and variable dataflow data, e.g. IP packets payload as in case of TBS-IP
- TVS implementation should support both in-band dataflow token-based signalling and control plane signalling using XML-based tokens
  - To allow in-band token-based signalling, token key and token should be of fixed length
- TVS should maintain own run-time table “token – GRI – (LRI) – (token key)”. Additionally The TVS table may contain a status or validity period of the tuple
  - GRI and/or LRI will link to actual local resource reservation table maintained by the resource reservation and management service and contain all necessary details
- TVS should allow smooth integration into more general AAA infrastructure and support multidomain resource reservation/authorisation