WP4: Authentication, Authorisation, Accounting (AAA)

WP4 Technical meetings

Yuri Demchenko <demch@science.uva.nl>
System and Network Engineering Group
University of Amsterdam

POSPHORUS General Assembly Meeting
7-9 April 2008, Barcelona

WP4 Internal Meeting

- WP4 internal meeting - 10:00-11:00 April 8 (Room A), next after WP1 meeting

1. WP4 M19-M30 plans and deliverables - YD

2. ForCES TBS and multilayer TBN - MC and EH

3. GAAA-TK components development for multidomain NRP - YD (including TVS, XACML policy, IBC and trust mngnt, configuration etc.)

4. UvA multidomain AAA testbed and SC08 scenarios - Discussion, FW and all

WP1-WP4 Meeting

- WP1-WP4 meeting - 11:30-12:30 April 8, 2008 (Room B)

  1. Discussion: GAAA/AuthZ library and interfaces
     1) Attributes for AuthZ
     2) Access control policy model, XACML implementation

  2. TVS and Pilot token for flexible NSP/NRPS integration

  3. SC08 Demo discussion

WP2-WP4 Meeting

- WP2-WP4 meeting - 9:30-11:00 April 9, 2008 (Room A)

  Goal: WP2-WP4 integration issues

  1. GAAA-TK pluggable components/library - YD
  2. WP2's vision on integrating AuthN/AuthZ services - WP2's TBD
  3. G2MPLS and TBN integration - MC
  4. Discussion: common middleware platform, interfaces, etc.

WP3-WP4 Meeting

- WP3-WP4 meeting - 8:30-9:30 April 9, 2008 (Room B)

  We need this meeting to discuss issues related to Grid/Unicore middleware integration and about some cooperation on Metascheduler.

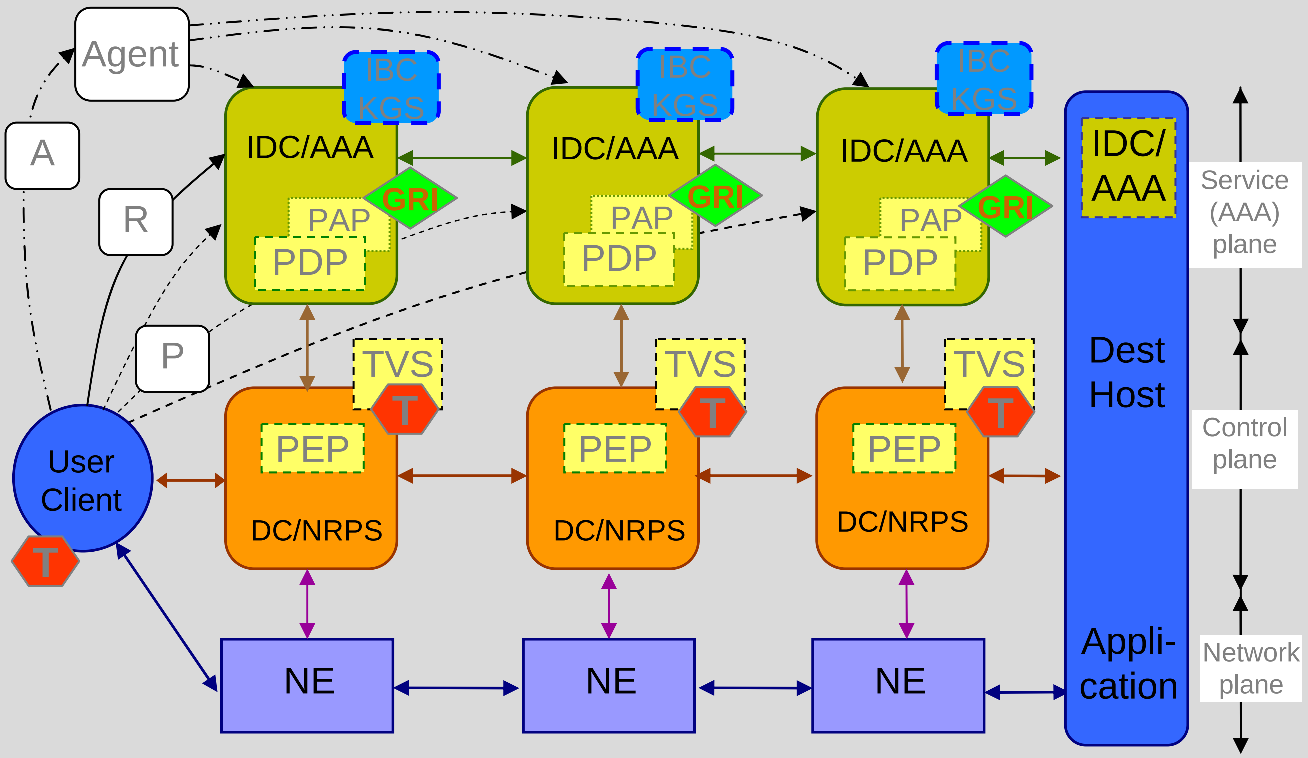
Outline – AAA/AuthZ infrastructure for ONRP

- AAA/AuthZ Architecture for Optical Network Resource Provisioning (ONRP)
  - "Provisioning – access" vs "provisioning – deployment - access"
- AAA/AuthZ functionality and GAAA Toolkit components to support ONRP
  - Interfaces and messages
  - Token Validation Service (TVS) and Token generation convention
  - XML token format a
- Using Identity Based Cryptography (IBC) for token key distribution at deployment stage
- Suggestions for SC08 Demo

Optical Network Resource Provisioning (ONRP)

- ONRP as a use case of the general Complex Resource Provisioning (CRP)
  - ONRP and Network on-demand provisioning
  - Grid Computing Resource – Distributed and heterogeneous
- 2 major stages/phases in ONRP/CRP operation
  - Provisioning consisting of 4 basic steps
    - Resource Lookup
    - Resource composition (including options)
    - (Advance) Component 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 reservation/AuthZ decision enforcement
- Now considering 2 stages "reservation-access" model vs 3 stages "reservation-deployment-access" model
  - Topic for WP4-WP1 and WP4-WP5 discussion

Multidomain Network Resource Provisioning (NRP)



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

- Token based policy enforcement
- GRI – Global Reservation ID
- AuthZ tickets for multidomain context mngnt
- T - Token

- NRPS – Network Resource Provisioning System
- DC – Domain Controller
- IDC – Interdomain Controller

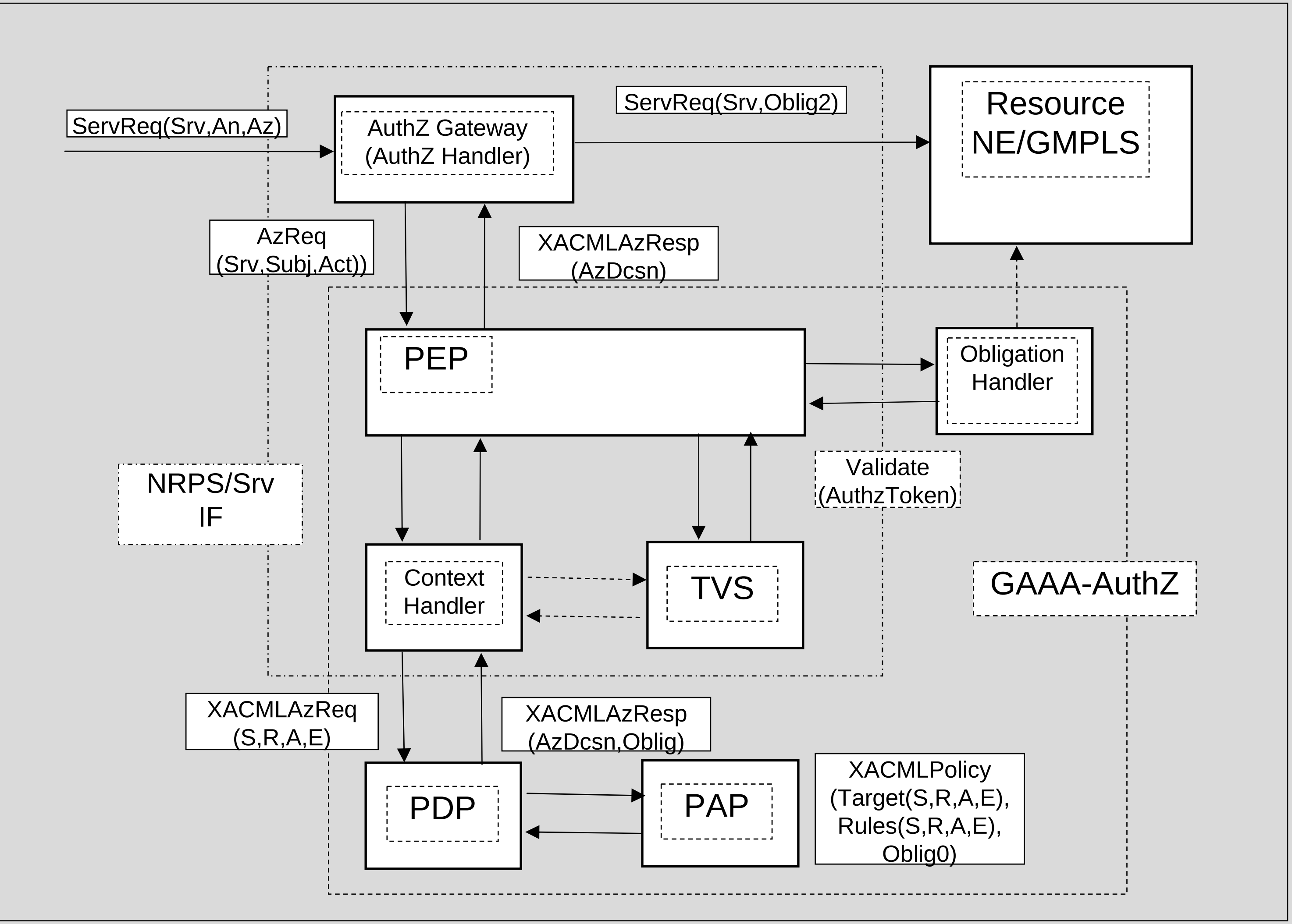
- AAA – AuthN, AuthZ, Accounting Server
- PDP – Policy Decision Point
- PEP – Policy Enforcement Point
- TVS – Token Validation Service
- KGS – Key Generation Service

AAA/AuthZ mechanisms and functional components to support multidomain ONRP

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 – To be discussed*
- AuthZ ticket 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 OLPP
  - Using reach functionality of the XACML policy format for complex network and Grid resources
  - *Potentially may use path/topology information – To be discussed with other WP's*
- *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

PEP-GAAAPI AuthZ Interface definition

- Method #1 - Returns Boolean value

```
Boolean authorizeAction (String resourceId, String actions, HashMap subjmap)
throws java.lang.Exception,
org.aaaarch.gaaapi.NotAuthenticatedException,
org.aaaarch.gaaapi.NotAvailablePDPException;
```

- Method #2 - Returns Boolean value

```
Boolean authorizeAction (String resourceId, String actions, String
   subjectId, String subjconfdata, String roles, String subjctx)
```

- Method #3 - Returns AuthZ ticket or token

```
String authorizeAction(String authzTicketToken, String sessionId, String
   resourceId, String actions)
```

- Method #4 - Returns AuthZ ticket or token

```
String authorizeAction (String authzTicketToken, String sessionId, String
   resourceId, String actions, HashMap subjmap)
```

Extracting AuthZ related information from Security/Message Context

MessageContext ((SubjCreds | SenderCreds), ResourceId?, Action?)

=> SecurityContext (SubjCreds, ResourceId, Action, Environment)

=> AuthzRequest (Subject, Resource, Action, (Environment))

- Function of AuthZ Gateway (AuthZ handler or interceptor)
  - Extract required information for AuthZ request from the message and application environment or context

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

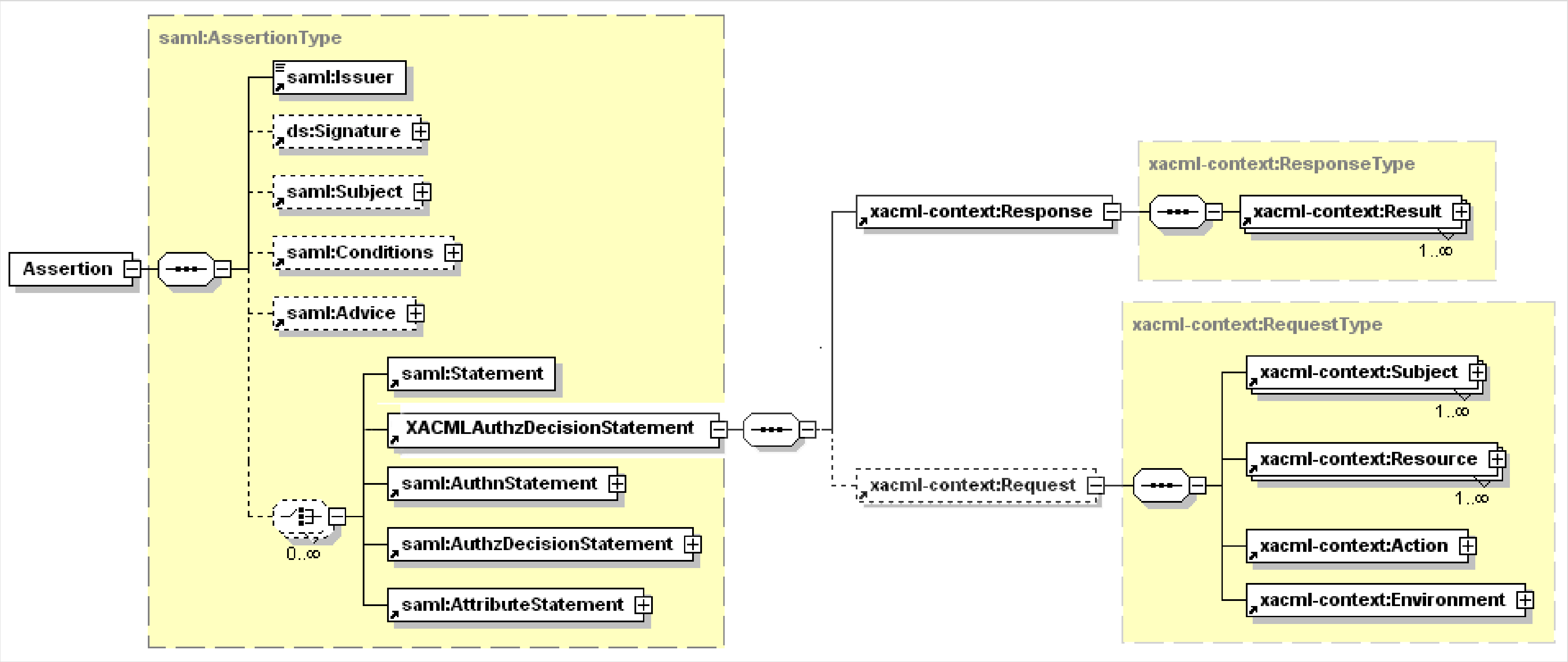
Response (Result (Status, Obligations)):

- 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

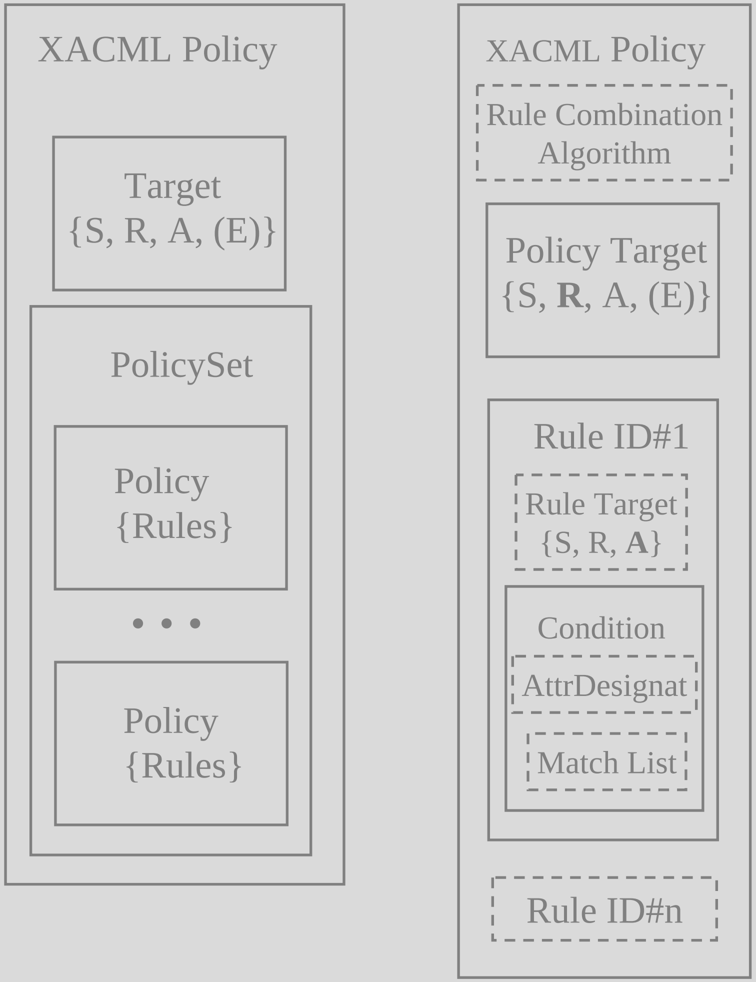


XACML Request-Response messages are enclosed into SAML2.0 Assertion SAML2.0 protocol messages

Extension library is available with OpenSAML2.0 and implemented in gLite and Globus TK4.1+

XACML Policy format

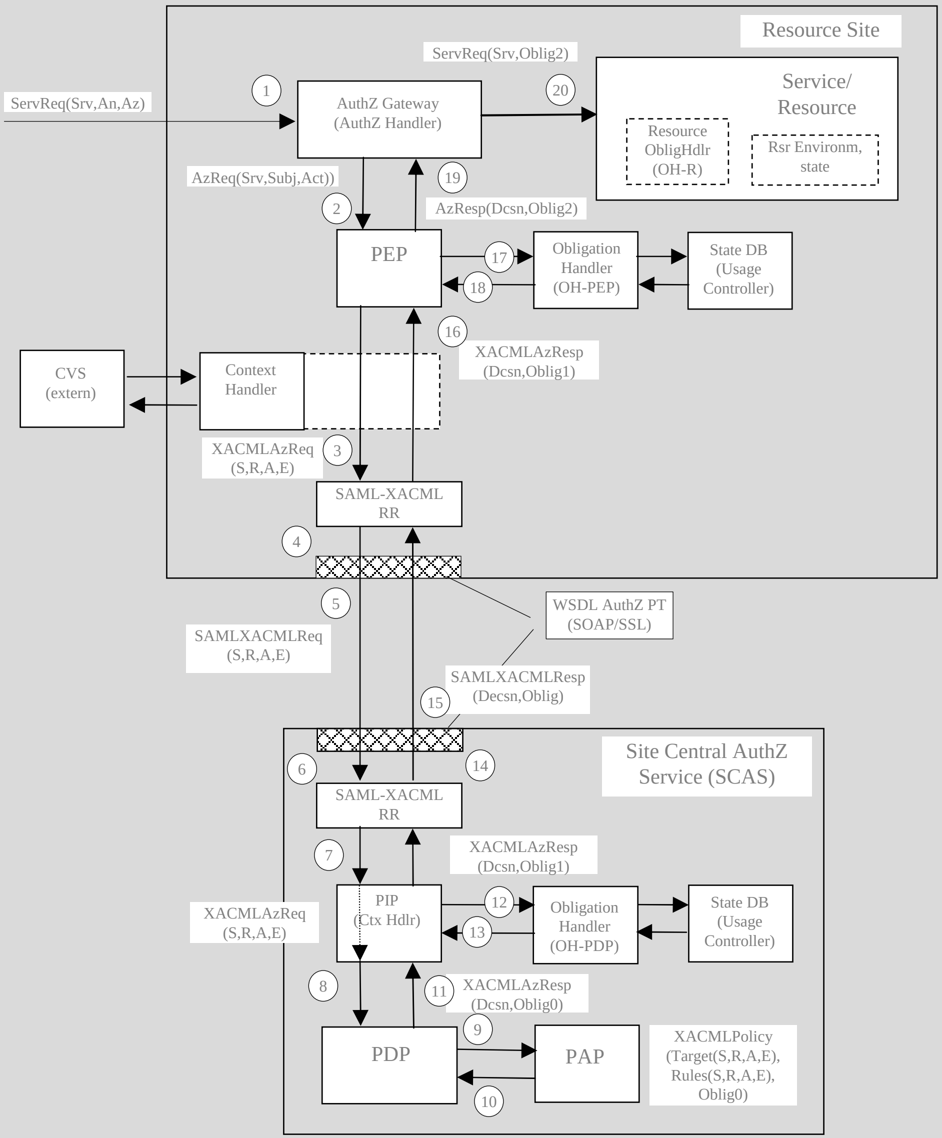
- Policy target is defined for the triad Subject-Resource-Action and may include Environment
- Policy may contain Obligation element that defines actions to be taken by PEP on Policy decision by PDP
- Obligations are part of PolicySet and Policy



Policy Obligations in Access Control and Management

- 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]
- Use in general Complex Resource Provisioning and Grid
  - Fixed, Time-flexible, Malleable/"Elastic" Scheduling
  - Account mapping, Quota assignment
  - Usable Resource
  - Accounting, Logging, Delegation
- 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
    - Require use of AuthZ assertions/tickets/(restricted proxy?)
  - Obligations are enforced before or after the resource or service accessed/delivered/consumed
    - Not discussed in current study/document – refer to OGSA AUTHZ-WG discussions

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.

GAAAPI Implementation and Configuration

- Implemented in Java (for IBC requires Java 6)
- Requires a number of supporting directories
  - Can be changed by modifying SecurityConfig class
- Can use pre-installed key-storage with private/public keys
  - To be a part of installation phase in future releases
- Special profile to support only TVS function and simple PEP function

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

Basic TVS functionality

- 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

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:TokenVal
   ue>
    AAA:Conditions NotBefore="2007-08-12T16:00:29.593Z"
   NotOnOrAfter="2007-08-13T16:00:29.593Z"/>
</AAA:AuthzToken>
```

where the element <TokenValue> and attributes SessionId and TokenId are mandatory, and the element <Conditions> and attributes Issuer, NotBefore, NotOnOrAfter are optional;

GRI = SessionId

TokenId – unique identifier (serving for logging and accountability)

- Binary token contains just two values – TokenValue and GRI

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:

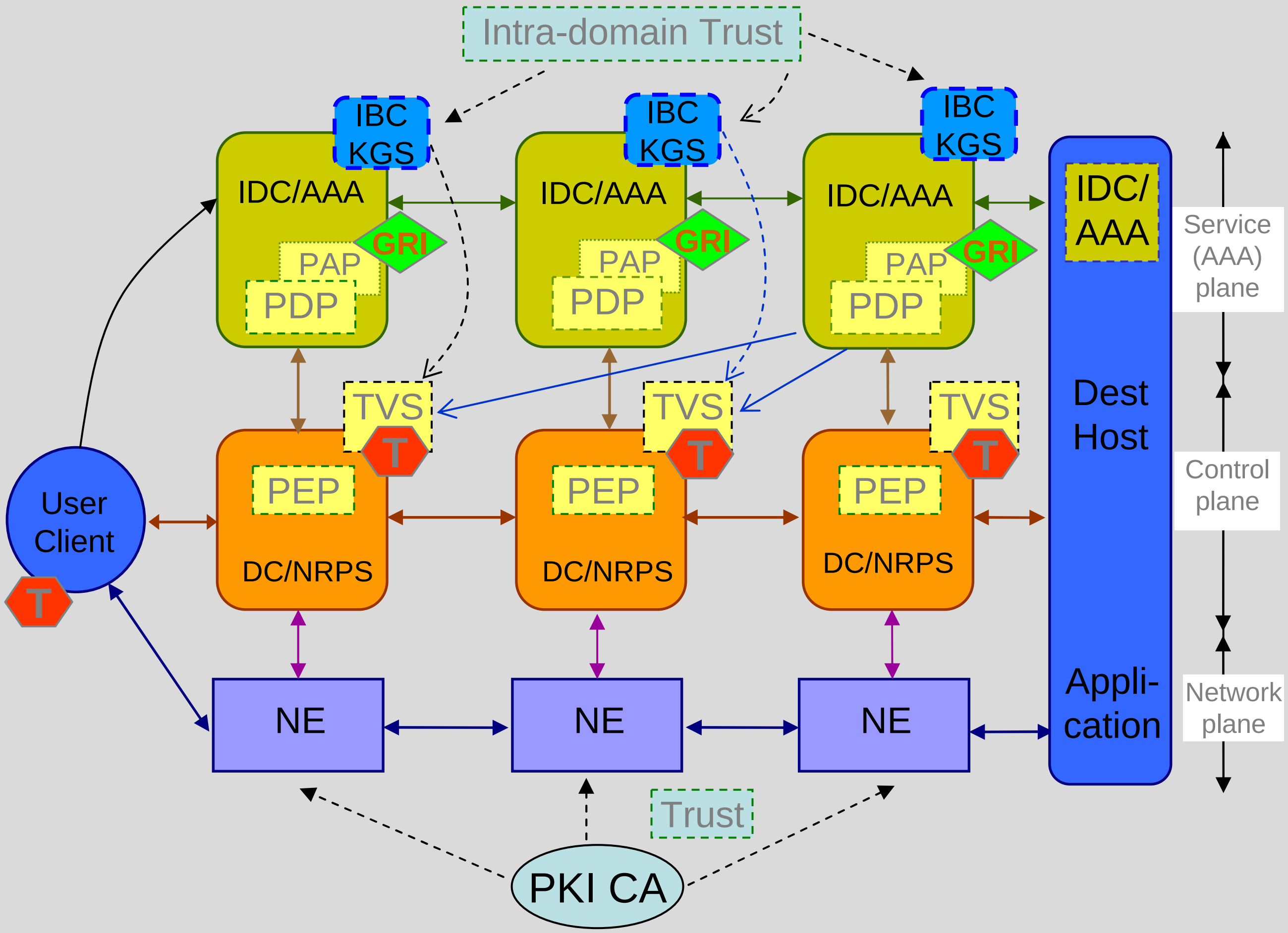
  `TokenValue = HMAC(GRI, TokenKey)`
- This algorithm allows for chaining token generation and validation process

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

Handling access tokens with TVS

- Using token for access control
  - 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

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

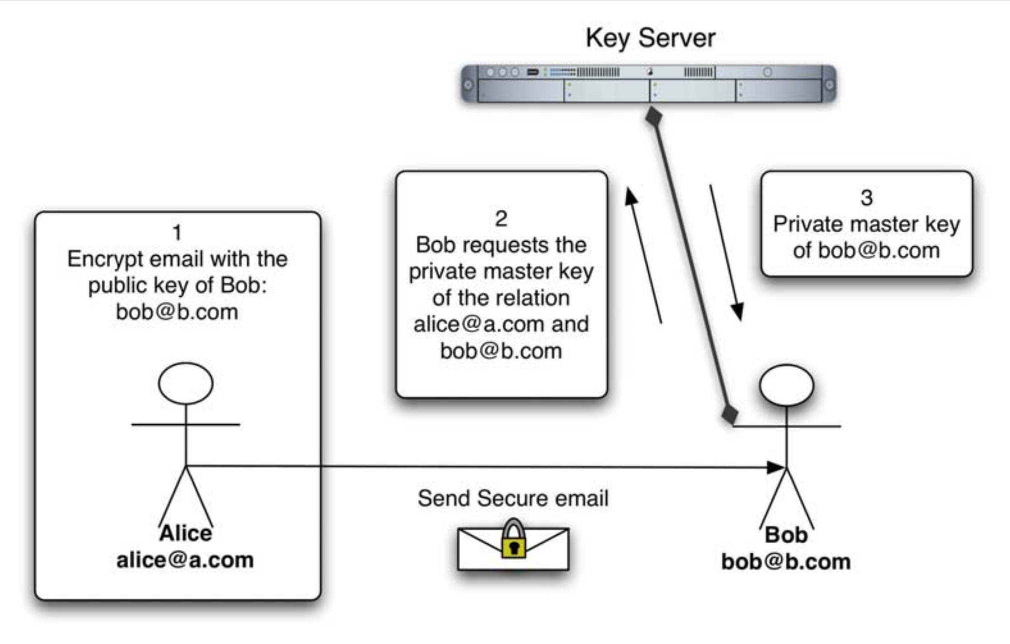
Used at deployment stage

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}

Suggestions for SC08 Demo

- ONRP/CRP model and supporting AAA/AuthZ infrastructure
  - Multidomain Lightpath Authorisation Architecture using Tokens
  - Using Pilot token at reservation stage
- Chain/Tree reservation/scheduling
  - Flexible scheduling and Advance reservation
- Using token for access control
  - More flexible comparing to AuthN/ID based approach
  - Separates reservation and access stages
  - Allows for multilayer token based access control
- Proposed and tested (in SC07) 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
- The required token handling functionality is supported by the TVS implementation
  - Planned to be extended to support IBC key distribution model

Discussion and Questions

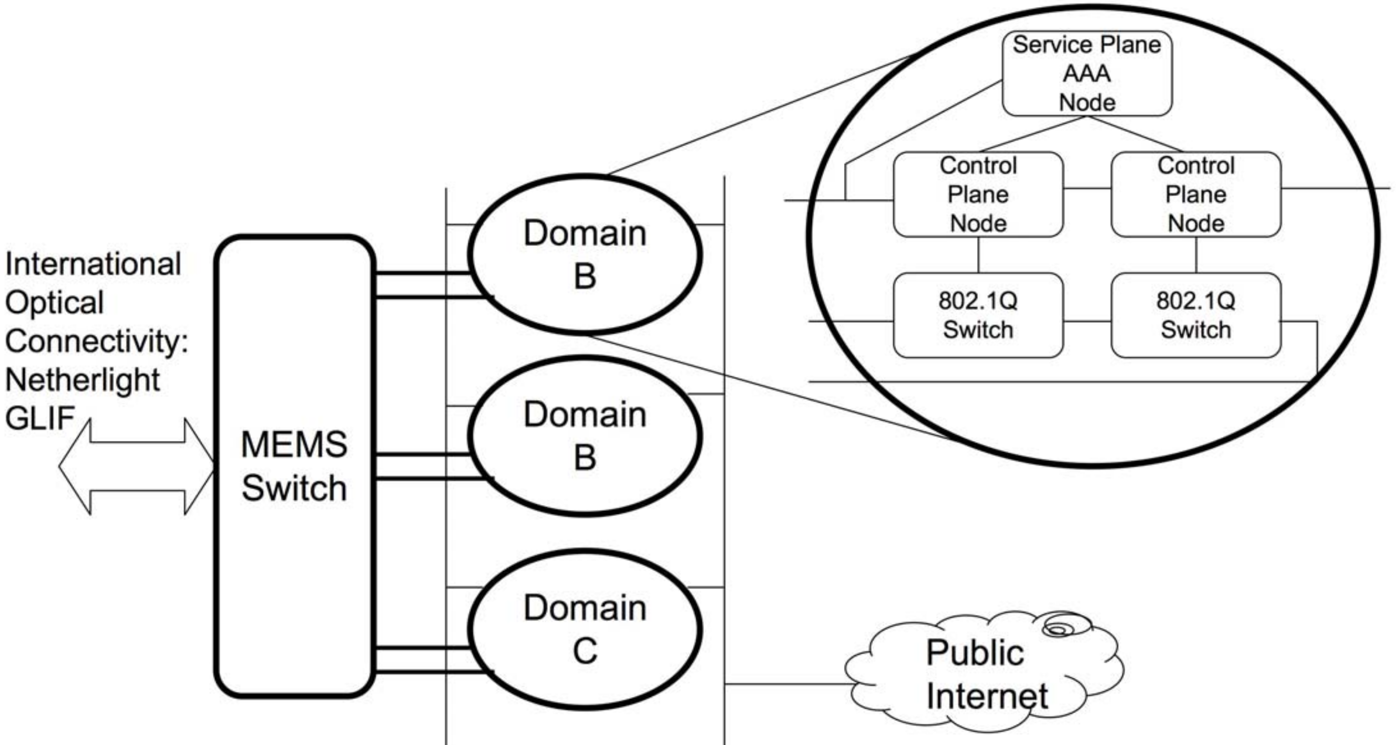
Additional materials

- Local UvA AAA testbed
- SC07 Demo

Local UvA Multi-domain AAA testbed

- Hosted by Amsterdam Lighthouse and contains 3 domains. Each domain consists of 3 CPU nodes
  - 2 nodes act as Control plane nodes, driving 802.1Q VLAN switches and accepting and forwarding signalling messages via an East-West interface and communicating operation and control messages via a North/South bound interface, which are generated by the 3rd node
  - The 3rd node acts as a Interdomain Controller (IDC) providing also AAA/AuthZ functionality for interdomain NRP
- It is intended to support various GMPLS implementations such as DRAGON and G2MPLS (when available from WP2)
  - NPRS based domains can be also implemented in the testbed
- Currently used for testing ongoing GAAA-AuthZ framework development for ONRP and being re-designed
  - Can be available to both partners from the Phosphorus project and organizations collaborating in the area of AAA, such as Internet2
- The UvA AAA testbed was used in the SC2007 Demo together with Internet2

Local UvA Multi-domain AAA testbed - Layout



Amsterdam Lighthouse
Multi-domain AAA testbed

3 domains consisting of

- 2 CP-nodes
- 1 SP/AAA node

SC07 Token Based Networking Demo

- Multidomain Lightpath Authorisation Architecture using Tokens
  - Tokens are a simple, fast and flexible way to authorize lightpaths
  - Tokens can be recognized by multiple domains
  - Tokens symbolize a commit of advance reservations by each domain
  - Tokens can be used at different layers in the network
  - Domains may or may not choose to enforce tokens (be transparent)
  - Allows separating complexity of authorization/reservation process from access or usage stage
  - Can support different accounting and billing models, e.g. pay-before (pre-pay) or pay-later (billing)
- Proposed and tested 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 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
- The required token handling functionality is supported by the TVS implementation

Reference - SC07 Token Based Networking Demo