Using WS-Federation and WS-Security for Identity Management in Virtual Organisations

Demchenko, Yu. <demch@science.uva.nl>, Universiteit van Amsterdam

Abstracts

The paper provides insight into one of key concept of Open Grid Services Architecture (OGSA) Virtual Organisation (VO) and analyses problems related to Identity management in VO and their possible solution based on using Ws-Federation and related WS-Security standards (including WS-Trust, WS-Policy, and WS-SecureConversation). The paper provides basic information about OGSA, OGSA Security Architecture and analyses VO security services. Some basic scenarios for WS-Federation use cases are described.

1. Computer Grids: Virtualisation of resources and organisations

Computer Grids [1, 2] emerged from the research area where the complexity of problems and need to share special unique resources required virtualisation of the collaborative environment but currently the Grid concept can also be used for typical business tasks where the large-scale dynamic resource sharing is a key problem. Grid technologies allow existing distributed computing to build dynamic cross-organisational applications and provide a standard base for new concepts in distributed utility computing and autonomic computing promoted by big computer vendors like IBM, HP and Sun [3].

All components of the Grid architecture are virtualised what allows creation of multiple virtual task oriented Grid Service instances on the same physical resources. Task oriented virtualisation of resources and services uses the concept of Virtual Organisation (VO). VO is defined as a set of individuals and institutions with belonging to them distributed resources [4].

VO is created to run specific (group of) tasks and may include multiple specially created Grid Services. VO is created on the base of the business agreement between participating organisations and individuals each of which contribute their specific resources (computers, services, people, etc.). The agreement defines all resources and services available to VO members and conditions on which these resources and services are provided and used. VO like real organisation may contain all basic services required to run typical organisation but these services “physically” and administratively may be run by member organisations on behalf of the VO. The examples of VO are: members of a large international long-term collaboration in high energy physics; or group of organisations participating in severe weather simulation and prediction; virtual laboratory involving group of specialists using remotely located unique analytical equipment (e.g., electronic microscope, or mass-spectrometer) for analysis of some samples.

Besides extensive requirements to high granularity of collaborative resource sharing, workflow management and underlying accounting functionality [1, 2], Grid applications require also specific functionality for VO management. VO membership management functionality extends beyond typical enterprise identity management concept and requires multi-institutional federation of people, resources and services.

Open Grid Services Architecture (OGSA) developed by joint effort of research community and industry in the framework of Global Grid Forum (GGF) is intended to create a standard base for building scalable VO’s and virtualising resource management [5, 6]. OGSA supports via standard interfaces and conventions the creation, termination, management and invocation of stateful transient services as named entities with dynamic, managed lifetime [grid book]. Within OGSA everything is presented as a Grid Service that conforms to a set of conventions for such purposes

as lifetime management, discovery of characteristics, notification and described as standard services in WSDL.

Virtualisation of resources in OGSA environment is based on Services Oriented Architecture (SOA) where any service is represented by common standard interface supported by specific Grid services (e.g., discovery, registry) to manage Grid service instance during its whole lifetime. Virtualisation of Grid services allows for consistent resource and services access across multiple heterogeneous platforms using common service semantic and interface mapped onto native platform.

2 OGSA (Open Grid Services Architecture)

Short overview of the Open Grid Services Architecture (OGSA) is provided here to show how is the concept of resources and tasks virtualisation is implemented and where is the place for Identity management in relation to VO. This will also be helpful for understanding what security services are required in OGSA and how they are related to VO functionality.

Computer Grids provide service oriented processing infrastructure incorporating distributed resource access and job execution. As an example, data object (or reference to persistent data location) together with bound job description (processing task) are travelling between computer systems enabling and relying on service negotiation and local resources management.

OGSA is being developed by Global Grid Forum (GGF) [4]. OGSA extends Web Services Architecture (WSA) [9] and provides framework for creating and managing stateful transient Grid Services. There are three principal elements of the OGSA Platform: the Open Grid Services Infrastructure (OGSI) [7, 8]; OGSA Services; and OGSA Platform Models.

The OGSA Platform Interfaces (and associated behaviors) are defined in OGSI in terms of WSDL (Web Services Description Languages) and provide necessary extensions to existing and emerging Web Services standards to support basic OGSA services. Separately from the basic OGSA components there will be defined profiles for protocol binding, platform binding and set of domain specific services.

2.1 Core OGSA Services

The OGSA document [6] describes a core set of services that appear as essential for many Grid systems and applications, and specifies at a high level the functionalities required for these core services and their interrelationship:

- *Discovery and brokerin*g mechanisms are required for discovering and/or allocating services, data, and resources with desired properties. Handle resolution that provides two level naming scheme for Grid services based on abstract long-lived GSH (Grid Services Handle) and less-long-lived GSR (Grid Service Reference).
- *Data services* that enable a service-oriented treatment of data, i.e. that data can be treated in the same way as other resources combined into virtual associations. Data management and sharing are common and important Grid applications. Mechanisms are required for accessing and managing data archives, for caching data and managing data consistency, for indexing and discovering data and metadata, and so on.
- *Security services* including infrastructure security and application related security services to provide identity and permission management, and security context management.

- *Agreements and Policy services* provide a general framework for creation, administration and management of policies and agreements for system operation, security, resource allocation, and in general provide infrastructure for policy-aware and policy enforcement services. It is important to be able to represent policy at multiple stages in hierarchical systems with a view to automating the enforcement of policies that might otherwise be implemented as organizational processes or managed manually.
- *Resource provisioning and management*, which encompasses service level agreement (SLA) negotiation, provisioning, and scheduling for a variety of resource types and activities.
- *Messaging and Queuing* to support a range of event messages for workflow and transactions management.
- *Monitorin*g that provide a global, cross-organizational view of resources and assets for project and fiscal planning, troubleshooting, and other purposes.
- *Logging, metering and accountin*g, which are required for security services, resource monitoring and billing, and used also for filtering, aggregation, persistency and configurable presentation.

The core OGSA services create a basis for other higher-level and more task oriented services that among others include workflow management, resource management and VO management services.

2.2 OGSI (Open Grid Services Infrastructure)

OGSI [7, 8] defines essential building blocks for distributed systems including standard interfaces and associated behaviors for describing and discovering service attributes, creating service instances, managing service lifetime, subscribing and delivering notification. A Grid service instance is a (potentially transient) service that conforms to a set of conventions expressed as WSDL interfaces, extensions and behaviors for such purposes as lifetime management, discovery of characteristics and notification.

Grid services are characterized (typed) by the capabilities that they offer. A Grid service implements one or more interfaces, where each interface defines a set of operations that are invoked by exchanging a defined sequence of messages. Grid service interfaces correspond to portTypes in WSDL. The set of portTypes supported by a Grid service, along with some additional information relating to versioning, are specified in the Grid service's serviceType, a WSDL extensibility element defined by OGSA.

Grid services can maintain internal state for the lifetime of the service. The existence of state distinguishes one instance of a service from another that provides the same interface. The term Grid service instance is used to refer to a particular instantiation of a Grid service.

Because Grid services are dynamic and stateful, dynamically created service instances must be distinguished one from another. For this purpose, every Grid service instance is assigned a globally unique name, the Grid service handle (GSH), that distinguishes a specific Grid service instance from all other Grid service instances that have existed, exist now, or will exist in the future. (If a Grid service fails and is restarted in such a way that it preserves its state, then it is essentially the same instance, and the same GSH can be used.)

Grid services may be upgraded during their lifetime, for example to support new protocol versions or to add alternative protocols. Thus, the GSH carries no protocol- or instance-specific information such as network address and supported protocol bindings. Instead, this information is

encapsulated, along with all other instance-specific information required to interact with a specific service instance, into a single abstraction called a Grid service reference (GSR). Unlike a GSH, which is invariant, the GSR(s) for a Grid service instance can change over that service's lifetime. Each GSR has an explicit expiration time, and OGSA defines mapping mechanisms for obtaining an updated GSR. Mapping is provided by Mapper invocating HandleMap GridService interface.

OGSI defines a class of Grid services that implements an interface that creates new Grid service instances. It is called the Factory interface and a service that implements this interface a factory. The Factory interface's CreateService operation creates a requested Grid service and returns the GSH and initial GSR for the new service instance.

Applications and users must be able to create transient services and to discover and determine the properties of available services. The OGSI Factory, Registry, GridService, and HandleMap interfaces support the creation of transient service instances and the discovery and characterization of the service instances associated with a VO.

3 OGSA Security Architecture and Services

OGSA security architectural components are required to support, integrate and unify available security models, mechanisms, protocols, platforms and technologies to enable a variety of systems to interoperate. Security services group encompass issues relating to the management and verification of credentials; privacy and integrity; and policy. OGSA Security WG at GGF underlined OGSA Security Architecture and Roadmap that defines OGSA Security Architecture, requirements to basic security services and relationships to WS-Security framework [14, 15].

OGSA Security Architecture defines all scope of services required to ensure end-to-end security of Grid services and applications: authentication, confidentiality, message integrity, policy expression and exchange, authorisation, delegation, single logon, credential lifespan and renewal, privacy, secure logging, assurance, manageability, firewall traversal, and OGSI layer security.

Establishing secure communication or context involves policy exchange and evaluation between service requestor and service provider. Policy can specify supported authentication mechanisms, integrity and confidentiality requirements, trust policies, privacy policies, and identity constraints. The security (and trust) model must provide a mechanism by which authentication credentials from the service requestor domain can be translated into the service provider domain, and trust relations are established.

Security domain for Grid services and applications may be defined by VO created on the base of agreement and establishing its own trust domain. VO members remain administratively independent and may continue running their own security services, the VO may provide a bridge for establishing trust relations between requestors and providers from different administrative and trust domains inside VO. The security model must provide a mechanism by which authentication credentials from the requestor domain can be translated into service provider domain.

OGSA Security architecture incorporates existing and emerging WS-Security standards and includes the following layers and components (see Fig. 1, from the bottom up):

1) Communication/transport Security Layer defines network infrastructure security and uses such network security services as SSL/TLS, IPSec, VPN, SASL, and others.

2) Messaging Security Layer is based on currently well defined and supported by different Web Services platforms SOAP/WS-Security. It also uses relevant XML Security mechanisms: XML

Signature, XML Encryption, and SAML and XACML security token exchange format. At this level security mechanisms are directly incorporated into OGSI services and definitions/formats.

3) Policy Expression and Exchange Layer defines set of policies applied to Grid Services and Grid operational environment which are required to ensure multi-domain and multiplatform compatibility. Policy layer provides necessary policy information for the Service/Operational Security layer. The proposed WS-Policy specification provides a framework to describe policies in a standard way and mechanism to include policies into service definition.

4) Services/Operational Layer defines security services/mechanisms for secure operation of Grid services in a open environment and includes:

- Secure Context Management
- Identity and Credential Translation and Federation
- Authorisation and Access Control Enforcement
- Auditing and Non-repudiation

Some of layers and components are described in more details below.

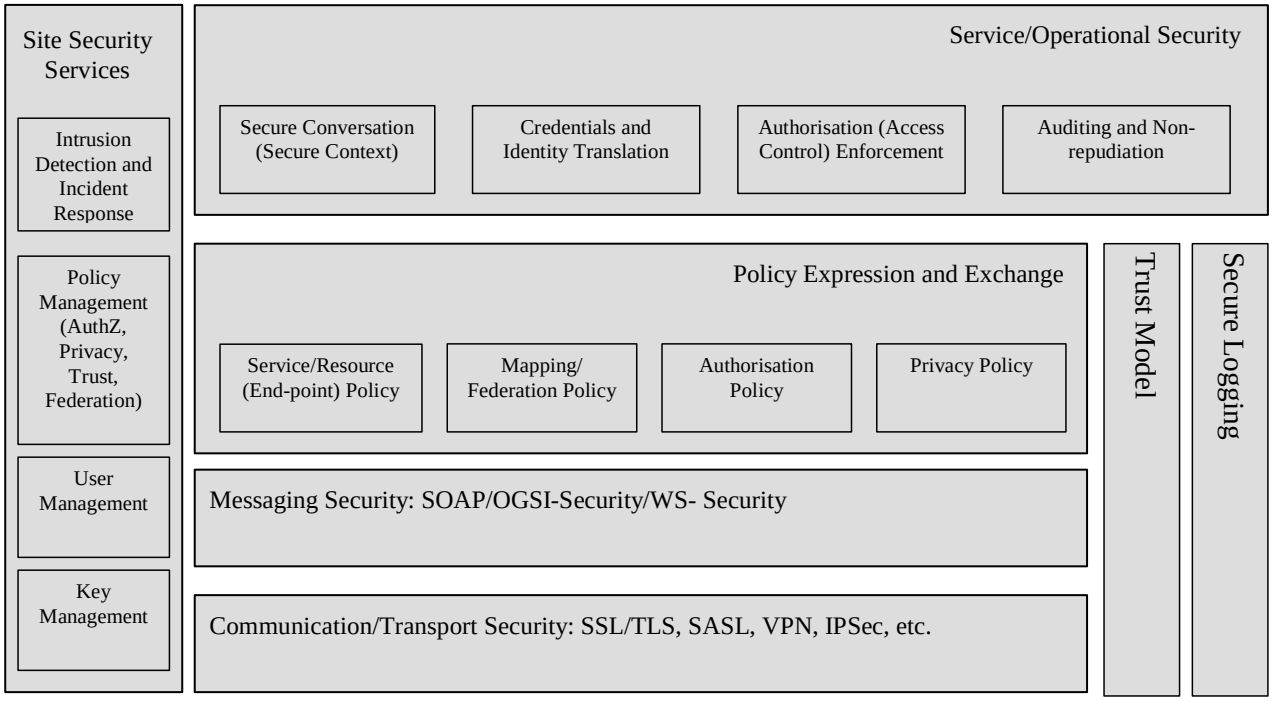

Fig. 1. Components of the OGSA Security Model.

a) Policy Expression and Exchange Layer

Interacting Grid services need to confirm to certain requirements in order to securely interact. It is important that service or resource requestors have access and understand policies associated with the target service. As a result, both the service requestor and service provider select acceptable security profile. It is also important to mention that the privilege to acquire Security Policy is given by the hosting environment to authenticated and authorised entities only.

Policy expression and exchange layer includes (but not limited to) the following policies:

- Local site policy and resource access policy, including VO policy
- Identity association/mapping and federation policy

- Trust policy, and
- Privacy policy

Policy layer provides necessary information for policy enforcement modules of the Service/Operational Security layer. It is suggested that policies expression should confirm to WS-Policy (and WS-SecurePolicy extension) that provides extensible framework that can be configured for specific applications based on several common attributes including privacy, security token requirements, token and other related information encoding, supported algorithms.

VO as a dynamically created entity requires the policy management services to provide a mechanism to distribute, negotiate and harmonise VO and local policies that may span multiple physical institutions and different administrative domains. VO Policy management concerns all policies related to the VO operation in the Grid environment.

Trust policy management provides a mechanism by which level of trust to the claims and assertions presented by others/entities is defined, and expressed in the Policies. Trust management issues are addressed by WS-Trust defined in the WS-Security.

Privacy policy management provides a mechanism to exchange and evaluate requestor and provider privacy policy to protect user anonymity or withhold private information.

b) Secure Context Management

Secure Conversation service adopts and leverages WS-SecureConversation specification to maintain consequent messages exchange between the Grid services that may span different VO's and over open network environment. Secure Conversation will maintain secure context established during initial mutual authentication for the period of active communication session between interacting application end points. Secure Conversation will operate at the SOAP message layer providing also binding with the policies associated with the end points.

c) Identity and Credential Translation and Federation

Grid services and applications typically span over multiple VO/locations and security domains that maintain their independent security services and policies. Operations between entities in different domains will require mutual authentication. Different security domains may incorporate different format and semantics for requestor/provider identities and credentials. Interoperation will require federation of the involved domains and identity and credentials translation or mapping. This federation may also be accomplished through trusted proxies or broker services. Identity mapping and federation is a subject to VO or local policies.

OGSA Identity specification will define how the identity name for an OGSA entity should be constructed based on the entity's identity established within their security domain. The specification considers cross-realm uniqueness, anonymity, and identity mapping. Other specifications will define cross-realm mapping for generic names, policy and credentials.

Specific for Web services and Grid services, delegation mechanism allows for a requestor to delegate some subset of their rights based on credentials delegation in order to fulfil the request. Delegation is based on credentials delegation by the authenticated entity and uses identity assertion profile to express identity assertion associated with a request, credential or communication context.

Identity and credential translation service can be built on two currently available identity management specifications WS-Federation (together with other complementary specifications WS-Trust, WS-Policy, WS-SecureConversation) and Liberty Alliance Project [17]. Identity management with WS-Federation is discussed below in details.

d) Authorisation and Access Control

Authorisation and Access Control security service is a key part of the managed security in an open service oriented environment. Authorisation is typically associated with a service provide or resource owner, who control access to a resource based on provided by requestor credentials or attributes that define requestor's privileges or roles bound to requestor's identity. Separation of Authentication and Authorisation services allows dynamic role based access control management and virtual association between interacting entities, and provides a basis for privacy in an open environment.

Authorisation and Access Control service in Grid applications/VO will re-use models proposed in WS-Authorisation that describes how access policies are specified and managed. Exchange of Authentication credentials and Authorisation attributes will be based on security token definition and exchange protocols defined in SAML and XACML [18, 19].

e) Auditing and Non-repudiation

Auditing and non-repudiation are necessary components for security services assurance and policy enforcement. They provide secure logging functionality that is required for many higher level audit related functionalities. Some limited auditing functionality may be required for other services at the Service/Operational Security level, in particular, timestamping.

f) Security Services Management

Effective and reliable operation of the security services requires underlying security services management and may include:

- key management for cryptographic functions;
- user management including user registry and related role or privilege management;
- policies management that includes local operational security policies, services security policies and trust management;
- intrusion detection and incident response capability.

These functions are related to local sites or VO's.

4. Virtual Organisation definition in OGSA

Virtual Organisation is defined in OGSA as a key concept for operation and managing Grid services [6]. VO supplies a context to associate users, resources, policies and agreements when making and processing requests for services related to a particular VO.

OGSA defines Virtual Organisation Management Service (VOMS) that provides functions for the creation and management of VO's including: a) registration and association of users and groups with the VO; b) management of user roles; c) association of services with the VO; d) associating agreements and policies with the VO and its component services.

Implementing and using VO concept requires a mechanism to reference the VO context and associate it with the user request. Creation of VO as a Grid services will allow using a standard

OGSI mechanism GSH (Grid Service Handle) that provides a unique persistent ID for VO as a Grid service.

VO membership service establishes and manages relationship of entities to the VO. These entities can be users, groups, other VO's or Grid resources. VO creation is initiated by establishing agreement between VO members that, among others, may include service providers and customers. In this way VO follows the same procedure as real organisation and in case of business oriented VO this stage may require relevant legal basis for establishing and operating such a VO. Web Services framework specifies necessary semantics, formats and protocols based on WS-Agreement, WS-Policy and other WS-Security extensions described below.

VO membership service is supported by the VO Registry that contains authoritative information about the entities and services associate with the VO, or VO's associated with the particular entity or user, and may also contain or reference user credentials and other related information. Although VO reference can be included into the request by the requestor, processing of the request may require obtaining authoritative information from the VO Registry. VO Registries should support different queries on any VO attribute or component, and also provide propagations of digest information between other Registries.

In order to securely process requests that traverse between members of a VO, it is necessary for the member organisations to have established trust relations. These trust relations may be direct or mutual, or established via intermediaries like VO Trust Management Service.

4.2 VO Security Services

VO can be established according to a well-defined procedure and based on common agreement between member organisations to commit their resources to the VO and adhere common policy that may be simple enough but not to contradict to the local security policies at member institutions. VO security policy cannot be more strict than the local policies, however if some specific enforcement is required by a specific VO, it should be also supported by local policies.

VO establishes own virtual administrative and security domains that may be completely separate or simply bridge VO members' security domains. This is required to enable secure service invocations across VO security domain but also require coordination with the security policies in member organisations.

The following security services and related functionalities are required for VO [15, 16]:

1) Trust management service
- Trust relations in VO may be built on the base of VO agreement or by means of PKI
- VO Agreement provides initial base for building trust relation inside VO
- VO maintains its own Certification Authority (CA) or provides a Bridge CA service

2) Policy Authorities
- Policy Authorities provide VO-wide policies related to authorisation, trust management, identity federation, mapping of identities, attributes and policies;
- Local Policy Authorities may also co-exist with the VO Policy Authority; in this case special policy should define relations between VO and local policies, including mapping rules between policies.

3) Identity Management Service
- Identity Provider (IP) service or Security Token Service (STS), that may also include

- Identity Federation service that provides federated identity assertions for users or resources.

4) Attribute Authorities
- Attribute Authorities can issue attributes bound to users or resources (represented by identities) that can be used for authorization decision when accessing VO resources or services.
- VO may also use local Attribute Authorities; in this case special policy should define mapping rules between VO attributes and local attributes.
- VO Attribute Authority may provide a Pseudonym service for the VO members.

5) Authorization service
- Authorization service enforces access control to a resource or service based on entity's attributes/roles and authorisation policies
- Authorisation service may be split on policy-based decision making module and policy enforcement module aligned to a specific resource or service.

VO can also have other services which are important for it's functioning such as logging, accounting, auditing/non-repudiation, etc. Physically, all VO services may be provided by member organisations on behalf of the VO and be distributed.

VO is normally created for a specific task, which however may be long lived. A VO lifecycle includes stages of creation, operation and termination. Initial stage of VO forming, after its establishing, also includes the following steps:

1) Establishing VO trust domain that defines trust relationships between VO members;
- VO Agreement provides initial basis for the trust relations between VO members;
- VO trust domain may be established on the base of PKI; two options are possible: VO can establish root Certifications Authority (VO-rootCA) to maintain hierarchical trust relations inside VO, or provide Bridge CA (VO-bridgeCA) function/service for member CA's

2) Establishing and/or configuring VO basic services, particularly: Identity Management service, Authorisation Service, Attribute Authority, Policy Authority. This work is normally performed by VO Administrator.

It is perceived that an initial VO is created by individuals or organisations (in fact, real or virtual, depending on agreement and policy) on the base of the Agreement between all member organisations that also defines a set of policies that VO must adhere. The following implies:

- VO may be created as a "close" VO with explicitly defined members however with the possibility to be extendable on base of individual participation, or as an "open" VO which membership is defined by specific roles or functions. As an example, in the first case, VO creates a special task-oriented infrastructure for cooperative use of the resources and services belonging to and provided by VO members; in the second case, VO may be created for providing some service or group of services that have already established infrastructure.
- VO Agreement forms VO administrative domain;
- Basic VO services are created at the initialization stage and may include Identity Management service, Authorisation Service, Attribute Authority, Policy Authority, however a subject to VO Agreement.
- Policies will drive all further steps of forming VO infrastructure, including adding and resigning new members, and define its operation;

- VO's may create hierarchical structure and create another VO's, however first or "initial" VO still must be created by human principal or real organisation as such VO foundation elements as agreement and policies are rather human-crafted;
- VO may be created as a Grid Service by a VO Factory service (using OGSI CreateService mechanism) that will also provide VO communication and messaging infrastructure and binding platform for higher-level services.
- VO administrator's role is created at the initialization stage and will have all privileges to form the initial VO structure and administer VO operation in the future. VO administrator will use available administration tools for the particular runtime environment, of which most of administrative functions and interfaces will be standardized in OGSI/OGSA and consequently automated to be run by authorised entities.

Figure 2 illustrates relationship between Virtual Organisation X and two member organisations A and B, which can be either real or virtual. VO X has been created to perform some task or provide some service(s) for designated group of users that constitute VO user community. VO established its own administrative and trust domain. Some of the users of organisations A and B may become the members of the virtual organisation; some of the services provided by member organisations may become VO services. Based on their obtained identity in the virtual organisation, VO users and services can interact in the trusted manner using VO security services, which in some operations may also request related services from the member organisations.

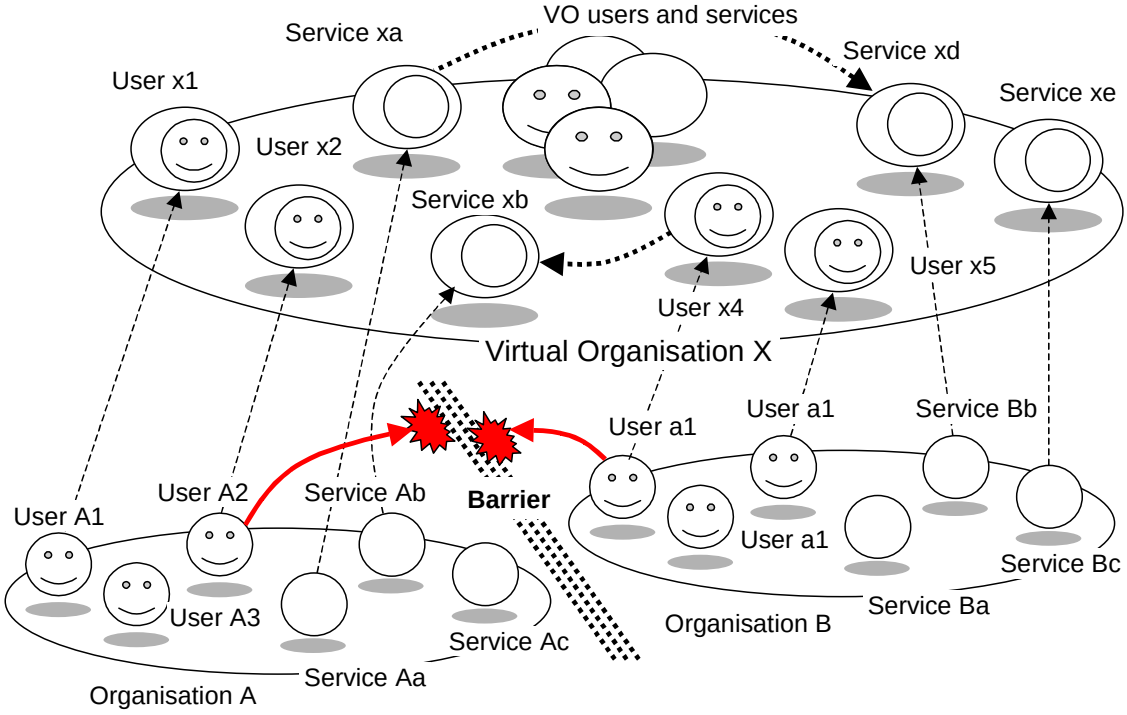

Figure 2. Virtual Organisation structure and relationship with member organisations

Figure 3 shows basic services representing VO security infrastructure that provides reliable, secure and accountable VO operation, and illustrates in details how the request from the service xa (that actually represents service Aa from the member organisation A) to the service xd (representing service Bb from the member organisation B) is handled using VO context. To become an active entity in the VO, a service or user must authenticate themselves to the VO Authentication Service (step 1). Then, the requestor service requests a security token from the Identity service (step 2), this step may also include obtaining specific attributes for the service xd from the Attribute

Authority (step 2a). Now the security token together with the obtained attributes may be presented to the target service xd (step 3). It's suggested that the resource will the trust security token issued by the VO's Identity service, otherwise it may verify presented credentials with the Identity service. Before granting or denying access, the resource may request VO Authorisation service to evaluate user request and presented credentials against access control policy that may be obtained from the Policy Authority step 4, 4a, 4b).

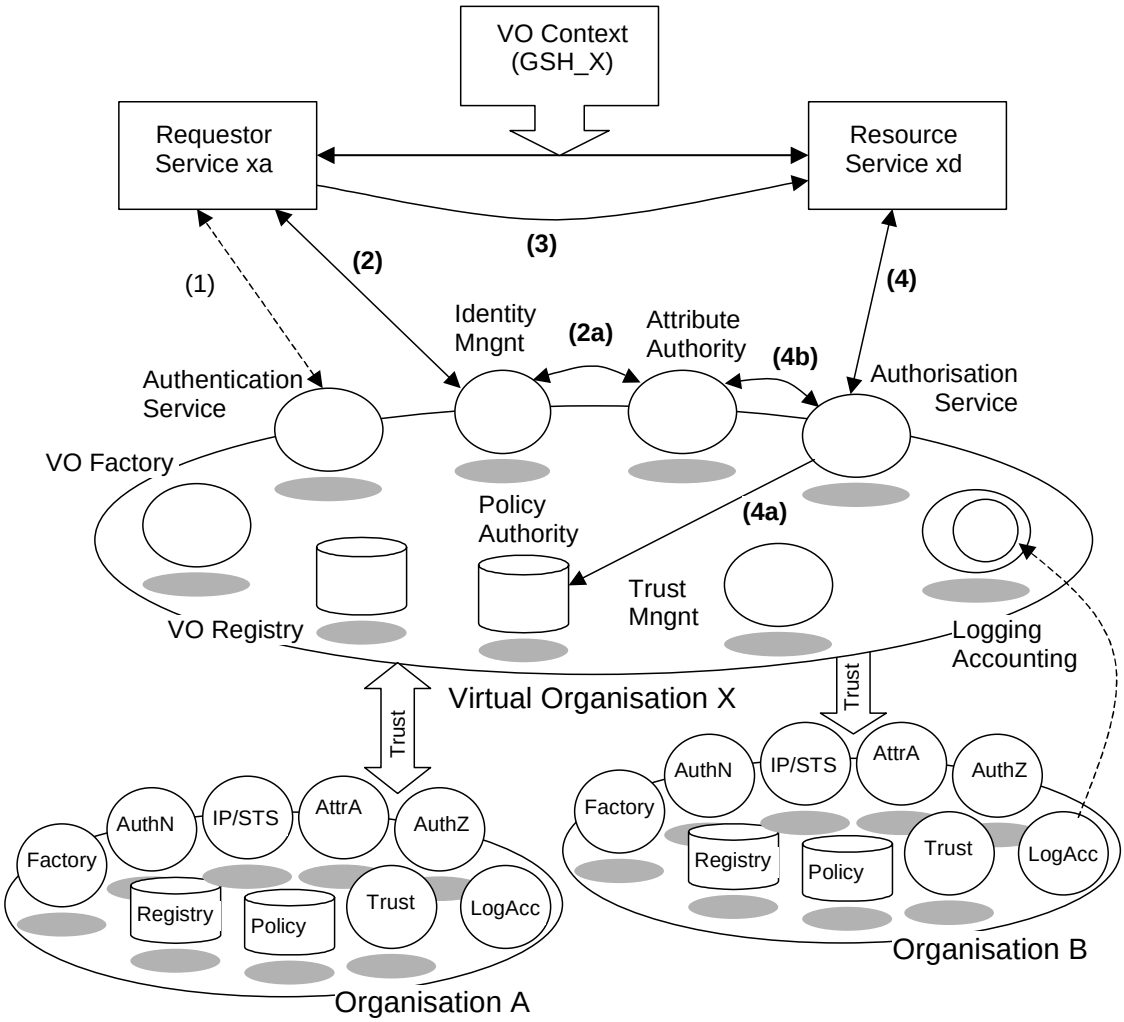

Fig. 3. Interaction of the VO security services when processing a request inside VO

5. Identity Management with WS-Federation

5.1 WS-Security framework

Addressing critical importance of the security for reliable operation of heterogeneous and multi-domain services and business processes, Web Services Architecture (WSA) [9, 10] offers extended architecture for Web Services Security (WS-Security) [22, 23].

WSA includes core specifications SOAP (Simple Object Access Protocol) and WSDL (Web Services Description Language) from W3C [WSA, WSDL] and UDDI (Universal Description, Discovery, and Integration) [11], which together provide service description, discovery and messaging framework for Web Services applications. Extended WSA includes such specifications as WS-Policy, WS-Coordination, WS-Transaction, WS-Inspection, WS-Addressing, and WS-

Security framework [23]. Some other components are added to the WSA framework cooperatively with the Grid community, in particular, WS-Agreement as a set of Web/Grid services to provide a framework for negotiating agreements [12], WS-Notification and WS-Resource Framework that add the ability to model stateful resources using Web services [13].

WS-Security describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies. WS-Security also provides a general-purpose mechanism for associating security tokens with messages and describes how to encode binary security tokens, in particular, X.509 certificates, Kerberos tickets, and encrypted keys. The WS-Security Profile for XML-based Tokens describes how to use XML-based tokens such as the SAML or XrML (the eXtensible rights Markup Language) with the WS-Security specification. It also includes extensibility mechanisms that can be used to further describe the characteristics of the credentials that are included with a message.

Other specifications from the WS-Security stack include WS-SecurityPolicy that specifies format for the policy assertions, and WS-Trust that enables Web Services to request and issue security tokens and to manage trust relationships. WS-SecureConversation defines extensions for secure communication by establishing and sharing security contexts, and deriving session keys from security contexts. WS-Federation introduces well-defined mechanisms and procedures for mapping trusted information about users from one domain into authentication and authorisation information required by resource or service provider from other domain. The functionality provided by WS-Federation is similar to identity federation functionality provided by Liberty Alliance Project [17] however it is more naturally integrated with other components of WS-Security framework.

5.2 WS-Federation Identity Management framework

WS-Federation defines mechanisms for federated identity management that are used to enable identity, attribute, authentication, and authorization federation across different trust realms [24-27]. The federation model extends WS-Trust model to describe how identity providers act as security token services and how attributes and pseudonyms can be integrated in security token mechanisms to provide federated identity. Tokens can represent the principal's primary identity or some pseudonym. Services can request attribute/identity service based on provided token/pseudonym to obtain authorised information about the identity. WS-Federation Active Requestor and Passive Requestor Profiles define how the cross trust realm identity, authentication and authorization federation mechanisms can be used by active requestors such as SOAP-enabled applications, or by passive requestors such as Web browsers to provide Identity Services.

Identity federation allows businesses to securely federate Web services to directly provide services for customers registered at other partner businesses or institutions within federation defined as a collection of domains that have established trust.

Virtual Organisation provides a good example where the identity and services federation are very important. VO can be presented as a federation of organisations (both real and virtual), resources and services provided by member organisations, and users affiliated to these organisations. Each of member organisations may maintain their own trust domain, resource and user identities. VO creates its own trust domain, identity and attribute services that may simply bridge trust relations and identities across VO. VO virtual services can be run by specially designated VO Grid Factory service or by one of member organisations. Therefore for Grid applications which are built using OGSI extensions to Web Services, WS-Federation and related WS-Security standards provide a native platform for establishing and running federated VO security services - trust relations,

identity and attribute authorities. WS-Federation framework can also address other issues in VO operation such as preserving autonomy of all members, respect and use pre-existing business and trust relations, protect individuals' privacy. As most of VO management and operation functions can be automated, there should be a set of policies defined for all trust and identity management functions.

5.3 Federated Identity Model

WS-Federation specification [25] defines a ***federation*** as a collection of realms that have established trust. The level of trust may vary, but typically includes authentication and may include authorization, or other actions or claims upon which one party is willing to rely on other party. A ***Trust Domain/Realm*** is an administered security space in which the source and target of a request can determine and agree whether particular sets of credentials from a source satisfy the relevant security policies of the target. The target may defer the trust decision to a third party (if this has been established as part of the agreement) thus including the trusted third party in the Trust Realm.

WS-Federation introduces the following services and definitions used in the federated identity model [25]:

Security Token Service (STS) - A *security token service* is a generic service that issues/exchanges security tokens using a common model and set of messages. STS makes assertions based on evidence that it trusts, to whoever trusts it. To communicate trust, a service requires proof, such as a security token or set of security tokens, and issues a security token with its own trust statement (e.g., co-signature). This forms the basis of trust brokering. A ***security token*** represents a collection of claims that may include name, identity, key, group, privilege, capability, attribute, etc.

Identity Provider (IP) - *Identity Provider* is an entity that acts as a peer entity authentication service to end users and data origin authentication service to service providers (this is typically an extension of a security token service).

Attribute Service - An *attribute service* is a Web service that maintains information (attributes) about principals within a trust realm or federation. The term principal, in this context, can be applied to any system entity, not just a person.

Pseudonym Service - A *pseudonym service* is a Web service that maintains alternate identity information about principals within a trust realm or federation. The term principal, in this context, can be applied to any system entity, not just a person. Pseudonym services can be provided by the Attribute Service.

Attribute/Pseudonym services can be used to provide mechanisms for restricted sharing of principal's information and identity mapping, which are a subject to the user authorisation and related policies.

Figure 4 illustrate a simple example how IP and STS from different trust domains interact to provide necessary credentials for a user/requestor to access a Resource. This example illustrates how the WS-Trust and WS-Federation models may be applied to simple federation scenarios [25].

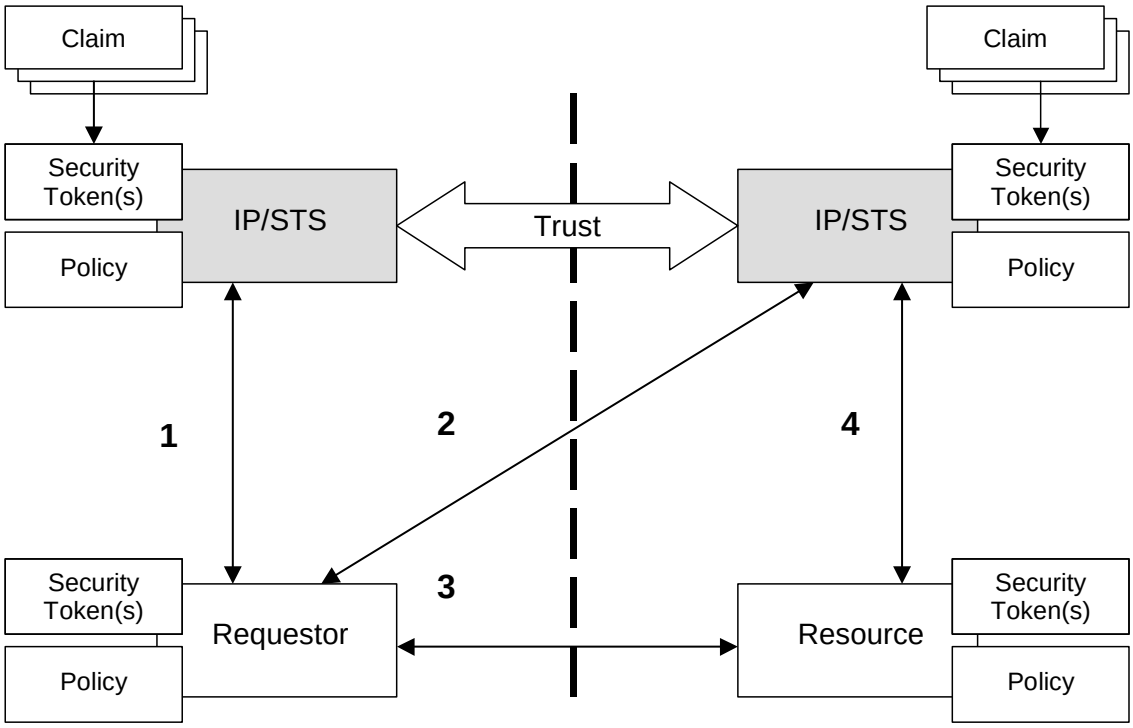

Figure 4. Interaction between IP/STS services in a simple federation scenario.

In this example a Resource requires that an incoming request prove a set of claims (e.g., name, VO/organisation, permission, capability, etc.) that may be used by the Resource authorisation service. A Resource represented by a Web services can indicate its required claims and related information as described by WS-Policy and WS-PolicyAttachment specifications [29, 30]. To access a Resource, a Requestor can include required security tokens into the message, apply signature to demonstrate a proof of possession, and send the message to a Resource. If the requestor does not have the necessary tokens to prove the claim, it can contact an appropriate IP/STS that may authenticate requestor's identity. For our example, the Resource requires the security tokens from its own trust domain. This means that the Requestor first needs to obtain the security token from Requestor's Identity provider that know Requestor's identity and can authenticate the Requestor and issues the authentication token. Now the Requestor can request the security token to access the Resource for the Resource's IP/STS service including already obtained authentication token into the request message. If the Requestor and the Resources have federated their trust domains, the Requestor's STS can issue the new security token that can be accepted by the Resource. These are steps (1) and (2) on our Figure 2.

When a Resource receives a request that contains security tokens (step (3) on Figure 2), it perform the following steps:

1) Verify that the claims are sufficient to comply with the policy and that the message conforms to the policy.

2) Verify that the attributes of the claimant are proven by the signatures (i.e., signed with the key trusted for the particular attributes) that also relates to the case when attributes were inserted by the trusted intermediary.

3) Verify that the issuers of the security tokens are trusted to issue the claims they have made; this may also require checking the chain of trust to the trusted, by a Resource, authority. The trust engine may also need to send security tokens to a STS in order to exchange them for other security tokens, which it can use directly in its evaluation.

If these conditions are met, and the requestor is authorised to perform the operation (what is a subject to authorisation decision by the Resource) then the Resource can process the request.

Depending on the authentication model and the policy, the Resource may accept the security tokens from the Requestor's domain (step (1) on Figure 4) and perform their validation with its own IP/STS service (step (4) on Figure 2), which will rely on previously established trust relationship with the Requestor IP service to validate the Requestor's security tokens and/or issue local security tokens.

WS-Federation and WS-Trust also enable other use cases where indirect trust relations are established via a trusted third-party, or a resource needs to access another resource on behalf of the initial requestor and will require delegation from the initial requestor. Delegation is an important mechanism for Grid applications and for enabling user initiated dynamic job/task execution.

Figure 5 illustrates a case when the Resource (B) needs to access another Resource (C) on behalf of the (original) Requestor (A) to perform the request. In this case the (second) Requestor (B) provides both delegation security tokens from the original Requestor and its own security tokens to the final service to ensure proper authorisation. This is a subject to the final service to verify security tokens and evaluate provided attributed to make the authorisation decision according to the resource policy. Delegation is defined by the delegation policy and may be restricted or un-restricted and normally is controlled by the user. Figure 5 also illustrates that there may be different combination of trust relations between the initial requestor and target resource, for example, there may be no trust relationship between domain B and domain C but there is a trust relationship between requestor's domain A and target resource's domain C.

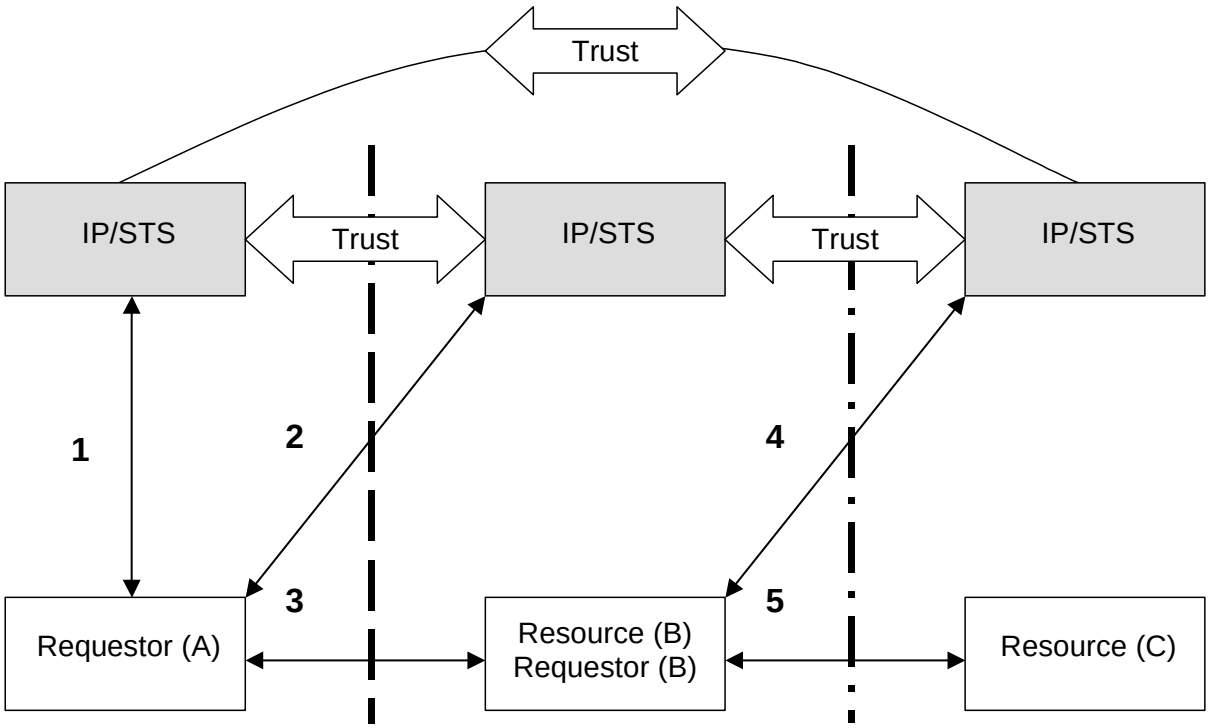

Figure 5. Delegation relationship

5.4 Attributes and Pseudonyms

WS-Federation specification extends the WS-Trust model to allow attributes and pseudonyms to be integrated into the token issuance mechanism to provide federated identity mapping mechanisms.

When requestors interact with resources in different trust realms (or different parts of federation), there is often a need to obtain some additional information about user in order to personalise user interaction, e.g. provide role/privilege based access level. A service known as an Attribute service may be available within a realm or federation and such a service can be used to obtain authorised information about a principal.

An Attribute Authority service is also necessary component of typical Role-Based Access Control (RBAC) [20] or Privilege Management Infrastructure (PMI) [21] that simplifies user roles and privileges management by storing them separately from the identity information. In multidomain applications authorisation services will require federation between multiple Attribute Authorities.

To facilitate single sing-on where multiple identity need to be automatically mapped and the privacy of the identity need to be maintained, there may be a pseudonym service. A pseudonym service allows a principal to have different aliases at different resources/services or in different realms, and optionally have the pseudonym change per-service or per-login. Figure 4 illustrates the general model for Attribute and Pseudonym services interaction. It is assumed that there is shared information or specialised trust to allow pseudonym service to perform the mapping or to make calls the IP to facilitate mapping.

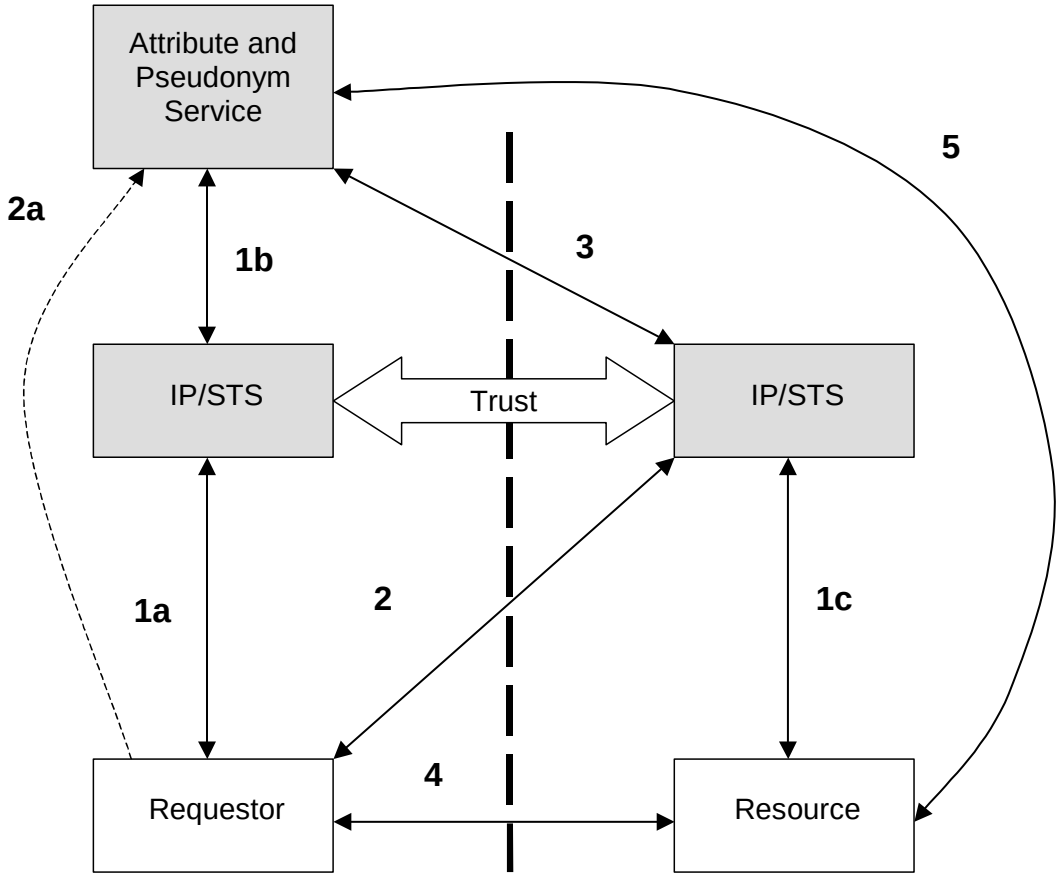

Figure 6. Interaction between Attribute/Pseudonym service and IP/STS

Similar to the basic federation scenario (as illustrated by Figure 4), the requestor obtains its identity token from its IP/STS (1a) and communicates with the resource's IP/STS (2). The resource IP/STS has registered a pseudonym with the requestor's pseudonym service (3). The requestor accesses the resource using the pseudonym token (4). The resource can obtain information (5) from the requestor's attribute service if authorized based on its identity token (1c). It should be noted that trust relationships will need to exist in order for the resource or its IP/STS to access the requestor's attribute or pseudonym service. In subsequent interactions, the requestor's IP/STS may automatically obtain pseudonym credentials for the resource (1b) if they are available. In such cases, steps 2 and 3 are omitted. Another possible scenario is that the requestor registers the tokens from step 2 with its pseudonym service directly (step 2a).

Pseudonyms are an optional mechanism that can be used by authorised cooperating parties/services to federate identities and securely and safely access profile attribute information. This is done by allowing services to issue pseudonyms for authenticated identities and letting authorized services query for profile attributes which they are allowed to access, including pseudonym specific to the requesting service.

In case there is a need to provide anonymity of tokens, pseudonyms provide a mechanism for ensuring this anonymity.

Figure 7 below illustrates one of scenario of using pseudonym to protect user privacy. The prerequisite to this scenario is the existence of the user/requestor pseudonym “Fredo” for the resource Fabrikam123.com. The next time the requestor signs in to Business456.com Identity Provider, they might be given a new identifier like "XYZ321@Business456.com". This is possible because the pseudonym service interacts with the IP and is authorized and allowed under the principal's privacy policy to perform this action. Since Business456.com Identity Provider received the mapping, the Web service at Fabrikam123.com can now request a local pseudonym for XYZ321@Business456.com and be returned "Freddo@Fabrikam123.com". The attribute service is able to do this because it has the ability to back-map "XYZ321@Business456.com" into a known identity at Business456.com which has associated with it pseudonyms for different realms.

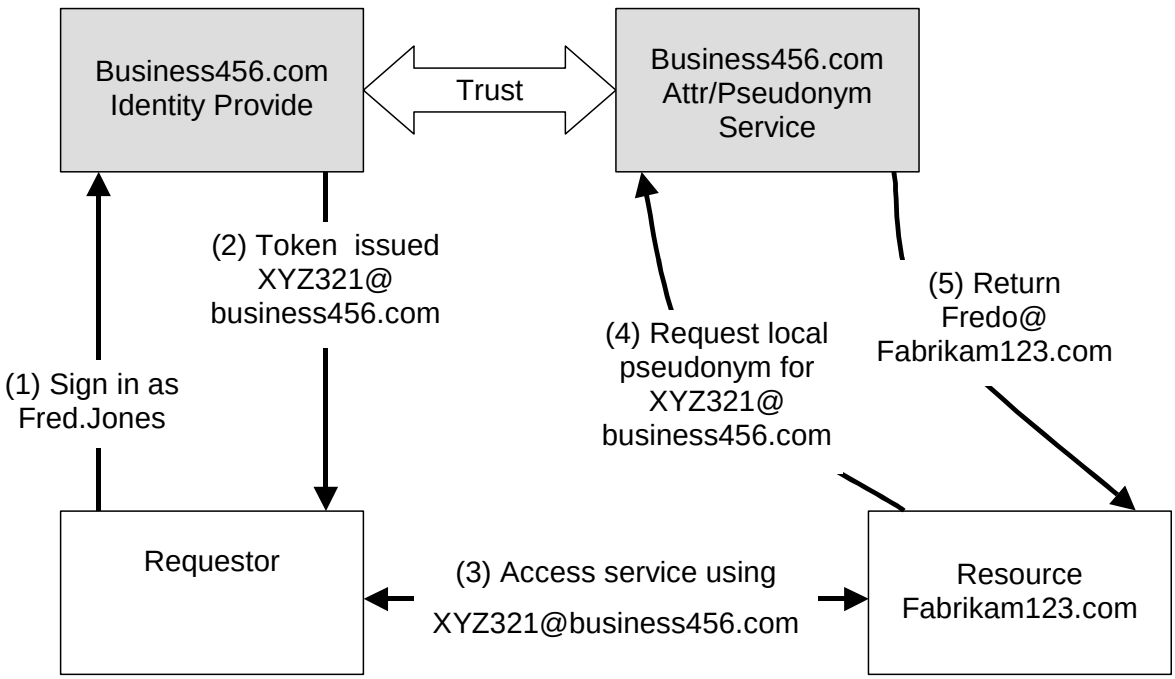

Fig. 7. Example of using pseudonym

Attribute/Pseudonym service gives a Requestor a possibility/mechanism to prevent identity tracking. The first approach is to have identity service to issue random ID/tokens each time a requestor signs in. In this case requestor receives the identity tokens that contains unique identifier, typically random. The resource or target services can request information about the requestor based on their pseudonym. The Pseudonym/Attribute service has the opportunity to enforce user privacy by releasing information only allowed by user and required for a specific service or action.

The pseudonym service is able to map-back provided pseudonymous token to the principal's known identity at the home domain/organisation. This process relies on the trust relationship between IP and Pseudonym service.

WS-Federation specification makes no proposals or requirements on the organisation of the data, just that a principal exists and can be addressable using described in the specification mechanisms. It is naturally that attribute and pseudonyms will use namespaces of affiliated organisations like in case of organisational LDAP directory, using URI format or naming similar to e-mail. It is also noted that each attribute may have different access, release and privacy policy, allowing principals to control to whom and how information is disclosed.

5.5 WS-Federation interfaces and messages

WS-Federation specification defines mandatory interfaces that must be supported by pseudonym service for getting, setting and deleting pseudonyms, and interface for Single Sign-Out service [25]. The specification contains WSDL file describing these interfaces as Web services methods.

WS-Federation uses standard for Web services SOAP messaging with WS-Security extensions for binding security assertions and context to the SOAP message. Below is the example of SetPseudonym message that sets the "Fredo" pseudonym of the principal identified by http://www.fabrikam123.com/MNK:

```
<S:Envelope>
  ...
  <S:Body>
    <wsse:SetPseudonym>
        <wsse:PseudonymBasis>
            <wsse:BinarySecurityToken>...<wsse:BinarySecurityToken>
        <wsse:PseudonymBasis>
        <wsse:RelativeTo>
          <wsa:Address>
                http://www.fabrikam123.com/NNK</wsa:Address>
        </wsse:RelativeTo>
        <wsse:Pseudonym>
          <wsse:SecurityToken>
              <wsse:UsernameToken>
                  <Username> "Fredo" </Userename>
              </wsse:UsernameToken>
          </wsse:SecurityToken>
          <wsse:ProofToken>...</wsse:ProofToken>
        </wsse:Pseudonym>
    </wsse:SetPseudonym>
  </S:Body>
</S:Envelope>
```

A sample response may simply contain empty message:

```
<S:Envelope>
  ...
  <S:Body>
    <wsse:SetPseudonymResponse/>
  </S:Body>
</S:Envelope>
```

It is important to note that the body of WS-Security compliant SOAP message contains (only) functional information, and all message security extensions, e.g. XML Signature, key information, are put in the message header.

a) Example WS-Trust messages for security token exchange

WS-Federation uses WS-Trust definitions (both metadata and messages) for security tokens issuance, validation and exchange, including requirements for delegation, forwarding and proxy [28]. Example below illustrates a message requesting X.509 security token based on the security token located in the `<Security>` header with the ID "myToken". This token specifies a username, and a signature is placed over the request using a key derived from the password (or password equivalent), nonce and timestamp.

```
<S:Envelope xmlns:S="..." xmlns=".../secext" xmlns:wsu=".../utility>
    <S:Header>
       ...
       <Security>
           <UsernameToken wsu:Id="myToken">
               <Username>NNK</Username>
               <Nonce>FKJh...</Nonce>
               <wsu:Created>2001-10-13T09:00:00Z </wsu:Created>
           </UsernameToken>
           <ds:Signature xmlns:ds="...">
               ...
           </ds:Signature>
       </Security>
       ...
    </S:Header>
    <S:Body wsu:Id="req">
       <RequestSecurityToken>
           <TokenType>wsse:X509v3</TokenType>
           <RequestType>wsse:ReqIssue</RequestType>
           <Base>
               <Reference URI="#myToken"/>
           </Base>
       </RequestSecurityToken>
    </S:Body>
</S:Envelope>
```

The following is a sample response containing custom security token along with an encrypted symmetric key for proof-of-possession.

```
    ...
    <RequestSecurityTokenResponse>
        <RequestedSecurityToken>
            <BinarySecurityToken ValueType="x:MyToken"
                                 EncodingType="wsse:Base64Binary"
                                 xmlns:x="...">
```

```
                MIIEZzCCA9CgAwIBAgIQEmtJZc0...
            </BinarySecurityToken>
        </RequestedSecurityToken>
        <RequestedProofToken>
            <xenc:EncryptedKey Id="newProof">
                ...
            </xenc:EncryptedKey>
        </RequestedProofToken>
    </RequestSecurityTokenResponse>
    ...
```

Delegation, forwarding and proxy requirements on the requested security token(s) are defined as extensions to the `<RequestSecurityToken>` element. The syntax for these extension elements is as follows:

```
    <RequestSecurityToken>
        <OnBehalfOf>...</OnBehalfOf>
        <DelegateTo>...</DelegateTo>
        <Forwardable/>
        <Delegatable/>
        <Proxiable/>
    </RequestSecurityToken>
```

b) Exchanging security context with WS-SecureConversation

Establishing federation or providing delegation means exchanging security context derived from the initial security token created by a security token service, by one of the communicating parties and propagated with a message, or security context token created through negotiation. WS-Federation and WS-Trust use for this purpose mechanisms for establishing and sharing security contexts, and deriving session keys from security contexts specified in WS-SecureConversation [32].

The following illustrates propagating a context to another party. Parties use already shared secrets to encrypt some sensitive information and sign the message body.

```
<S:Envelope xmlns:S="..." xmlns=".../secext" xmlns:wsu=".../utility>
    <S:Header>
       ...
    </S:Header>
    <S:Body>
        <RequestSecurityTokenResponse>
            <RequestedSecurityToken>
                <wsse:SecurityContextToken>
                    <wsu:Identifier>uuid:...</wsu:Identifier>
                </wsse:SecurityContextToken>
            </RequestedSecurityToken>
            <RequestedProofToken>
                <xenc:EncryptedKey Id="newProof">
                    ...
                </xenc:EncryptedKey>
            </RequestedProofToken>
        </RequestSecurityTokenResponse>
    </S:Body>
</S:Envelope>
```

c) Example WS-Policy Extensions

Policies associated with the security tokens or operations can be included and/or referenced via special extensions to the <RequestSecurityToken> element. The syntax for these extension elements is as follows:

```
<RequestSecurityToken>
    <wsp:Policy>...</wsp:Policy>
    <wsp:PolicyReference>...</wsp:PolicyReference>
</RequestSecurityToken>
```

Policy itself in the WS-Policy format can be represented by policy expressions containing policy assertions and limited number of operators [29-31]. Every assertion contains mandatory attribute wsp:Usage with possible value "wsp:Required", "wsp:Rejected", "wsp:Optional", "wsp:Observed", or "wsp:Ignored". Policy operators allow alternative acceptable policy assertions to be specified:

- <wsp:All> which requires that all of its child elements be satisfied
- <wsp:ExactlyOne> which requires that exactly one of its child elements be satisfied
- <wsp:OneOrMore> which requires that at least one of its child elements be satisfied
- <wsp:Policy> whose semantics are the same as <wsp:All>

The following example illustrates the representation of a group of assertions as part of an expression with an <wsp:ExactlyOne> operator. In this example, the security profile consists of security specific policy assertions. The first alternative specifies Kerberos Authentication and Privacy. The second alternative specifies password authentication and an audit trail. Either the first or second alternative must be chosen, but not both. The preference attribute establishes a preference for the first alternative.

```
<wsp:Policy xmlns:wsse="..." xmlns:wssx="...">
  <wsp:ExactlyOne>
    <wsp:All wsp:Usage="wsp:Required" wsp:Preference="100">
      <wsse:SecurityToken>
        <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
      </wsse:SecurityToken>
      <wssx:Privacy />
    </wsp:All>
    <wsp:All wsp:Preference="1" wsp:Usage="wsp:Required">
      <wsse:SecurityToken>
        <wsse:TokenType>wsse:UsernameToken</wsse:TokenType>
      </wsse:SecurityToken>
      <wssx:Audit />
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

6. Summary

WS-Federation specification extends the WS-Trust model to allow attributes and pseudonyms to be integrated into the token issuance mechanism to provide federated identity mapping mechanisms.

Principals can request tokens for resources/services using the mechanisms described in WS-Trust and issued tokens may either represent the principals' primary identity or some pseudonym appropriated for the scope/target service. Tokens may also include policy assertions in the format of WS-Policy and WS-SecurityPolicy.

Identity Providers and Security Token Services (IP/STS) may communicate between each other depending on established trust relations. Principals are associated with their "home" real or virtual organisation's IP and attribute/pseudonym services that maintain their attributes and pseudonyms.

Attribute and pseudonym services can use any kind of repositories that for WS-Federation compliant services may be exposed as UDDI endpoints.

WS-Federation specification defines mandatory interfaces that must be supported by pseudonym service for getting, setting and deleting pseudonyms. WS-Federation uses standard for Web services SOAP messaging with WS-Security extension for binding security assertions and context to the SOAP message. WS-Federation uses security tokens and interfaces defined in WS-Trust, WS-Policy assertion about applied security policies, and WS-SecureConversation mechanisms for establishing and sharing security context and deriving session key from the security context.

WS-Federation defines Single Sign-Out as an important operation to terminate a federation which action should be to flash immediately any state or cached information related to a particular principal (identity or pseudonym). Single sign-out is pushed through federation by IP/STS on principal's sign-out request.

WS-Federation provides functionality similar to Liberty Alliance Project (LAP) Identity management and Single-Sign-On services. Like LAP, WS-Federation enables Identity and Single-Sign-On services for existing (pre-defined) business and trust relations between service providers and identity providers. However, Liberty federation model is rather user-centric but WS-Federation is initially more services oriented.

WS-Federation and LAP can be treated as competing standards, however there is more space for their coexistence and integration. Both standards are using WS-Security mechanisms for secure messages exchange. WS-Federation is naturally integrated into WS-Security framework and uses security mechanisms and interfaces defined in other WS-Security specifications (in particular, WS-Trust, WS-SecureConversation, WS-Policy). LAP, which is based on SAML extensions, is more independent from underlying messaging platform and context exchange mechanisms, however this also brings more complexity to SAML based Liberty protocol.

Virtual Organisation is defined in OGSA as a key concept for operation and managing Grid services. VO supplies a context to associate users, resources, policies and agreements when making and processing requests for services related to a particular VO. VO can be established according to a well-defined procedure and based on agreement between member organisations (both real and virtual) to commit their resources and adhere common policies. VO requires a set of common security services that federate users, resources and services across VO security and administrative domains. WS-Federation and related standards can provide a native platform for VO identity management service, enabling VO members to request special identity/attribute service to invoke and control their identity at a specific location.

Reference

1. Grid Computing. Making the Global Infrastructure a Reality. Edited by F. Berman, G.Fox, T.Hey. – Wiley, 2003. - 1012 pp.
2. The Grid 2: Blueprint for a New Computing Infrastructure. Edited by I.Foster, C.Kesselman. – Elsevier, 2003. – 748 pp.
3. Utility Computing - http://utilitycomputing.itworld.com/

4. Anatomy of Grids - http://www.globus.org/research/papers/anatomy.pdf
5. Open Grid Services Architecture Working Group - https://forge.gridforum.org/projects/ogsa-wg
6. The Open Grid Services Architecture - https://forge.gridforum.org/projects/ogsa-wg/document/draft-ggf-ogsa-spec/en/13/draft-ggf-ogsa-spec.doc
7. Open Grid Services Infrastructure (OGSI). Version 1.0 - http://forge.gridforum.org/projects/ogsi-wg/document/Final_OGSI_Specification_V1.0/en/1/Final_OGSI_Specification_V1.0.pdf
8. Open Grid Service Infrastructure Primer (Draft) - https://forge.gridforum.org/projects/ogsi-wg/document/draft_ggf_ogsi_gridserviceprimer-recent.doc
9. Web Services Architecture, W3C Working Draft 8 August 2003 - http://www.w3.org/TR/ws-arch/
10. Web Services Description Language (WSDL) 1.1. 3C Note 15 March 2001 - http://www.w3.org/TR/wsdl
11. UDDI Version 3.0.1 - http://uddi.org/pubs/uddi-v3.0.1-20031014.pdf
12. WS-Agreement Structure, Version 0.1. - 9 January 2004 - http://www-unix.mcs.anl.gov/~keahey/Meetings/GRAAP/WS-Agreement%20Structure.pdf
13. The WS-Resource Framework - http://www-fp.globus.org/wsrf/default.asp
14. Security Architecture for Open Grid Services - https://forge.gridforum.org/projects/ogsa-sec-wg/document/Security_Architecture_for_Open_Grid_Services/en/2/Security_Architecture_for_Open_Grid_Services.doc
15. OGSA Security Roadmap - https://forge.gridforum.org/projects/ogsa-sec-wg/document/OGSA_Security_Roadmap/en/1/OGSA_Security_Roadmap.doc
16. OGSA-WG. Security Use Cases - Takuya Mori - https://forge.gridforum.org/docman2/ViewProperties.php?group_id=42&category_id=623&document_content_id=1717
17. Liberty Alliance Phase 2 Final Specifications - http://www.projectliberty.org/specs/
18. Security Assertion Markup Language (SAML) v1.0 - OASIS Standard, 5 November 2002 - http://www.oasis-open.org/committees/documents.php?wg_abbrev=security
19. eXtensible Access Control Markup Language (XACML) Version 1.0 - OASIS Standard, 18 February 2003 - http://www.oasis-open.org/committees/documents.php?wg_abbrev=xacml
20. Role Based Access Control (RBAC) – NIST, April 2003. - http://csrc.nist.gov/rbac/
21. ITU-T Rec. X.812(1995) | ISO/IEC 10181-3:1996, Information technology - Open systems interconnection - Security frameworks in open systems: Access control framework.
22. Security in a Web Services World: A Proposed Architecture and Roadmap - http://www-106.ibm.com/developerworks/webservices/library/ws-secmap/
23. Web Services Security Framework by OASIS - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
24. Federation of Identities in a Web Services World A Joint Whitepaper from IBM Corporation and Microsoft Corporation Version 1.0 July 8, 2003 - http://msdn.microsoft.com/ws-federation/
25. Web Services Federation Language (WS-Federation) Version 1.0 - July 8 2003 – http://msdn.microsoft.com/ws/2003/07/ws-federation/
26. WS-Federation: Active Requestor Profile. Version 1.0 July 8, 2003 - http://msdn.microsoft.com/ws/2003/07/ws-active-profile/
27. WS-Federation: Passive Requestor Profile. Version 1.0 July 8, 2003 - http://msdn.microsoft.com/ws/2003/07/ws-passive-profile/
28. Web Services Trust Language (WS-Trust) Version 1.0 December 18, 2002 - http://msdn.microsoft.com/ws/2002/12/ws-trust/
29. Web Services Policy Framework (WS-Policy). Version 1.1 28 May 2003 - http://msdn.microsoft.com/ws/2002/12/Policy/

30. Web Services Policy Attachment (WS-PolicyAttachment) - - http://msdn.microsoft.com/ws/2002/12/PolicyAttachment
31. Web Services Security Policy Language (WS-SecurityPolicy). Version 1.0. December 18, 2002. - http://msdn.microsoft.com/ws/2002/12/ws-security-policy/
32. Web Services Secure Conversation Language (WS-SecureConversation). Version 1.0 December 18, 2002. - http://msdn.microsoft.com/ws/2002/12/ws-secure-conversation/
33. Interoperability Prototype for Liberty - http://wwws.sun.com/software/sunone/identity/ipl/
34. SourceID: Open Source Federated Identity Management - Liberty Alliance, SAML, and WS-Federation - http://www.sourceid.org/wiki/Wiki.jsp