Towards cross-middleware authentication and single sign-on for ARC grid middleware

Weizhong Qiang
Department of Physics
University of Oslo
Oslo, Norway
weizhong.qiang@fys.uio.no

Aleksandr Konstantinov
Department of Physics
University of Oslo
Oslo, Norway
aleksandr.konstantinov@fys.uio.no

***Abstract*—In task of making the access to Grids as simple as possible, security is one of the most important challenges in production Grid infrastructures, especially when the Grid applications span across multiple administration domains as well as across heterogeneous Grid middlewares, such as some wide scale eScience applications which need to coordinate resources sharing among collection of autonomous institutions with different Grid middlewares running on these resources. In this paper, we describe the security implementation and consideration in new version of Advanced Resource Connector (ARC) middleware, where the heterogeneity issue have been exploited. The main goal of ARC implementation in terms of security is to let the middleware be capable of interoperating with other Grid middlewares by leveraging standard specifications. The key aspect of the work we have done is to enhance the current proxy certificate based authentication and single sign-on by utilizing and enhancing the standardized Web Service specifications such as Security Assertion Markup Languages (*SAML*) single sign-on (SSO) profile, Web Services Security in order to achieve cross-middleware authentication and single sign-on.**

Keywords- single sign-on; delegation; virtual organization; ARC middleware

I. INTRODUCTION

Grid provides the technology for wide-scale, cross-domain collaboration where the users work on resources and sharing data which are distributed across domains. This kind of collaboration is called virtual organization (VO)[1]. The resources that contribute to the VO could come from independent administrative domains, and even require different security approaches for control the access to themselves because different middlerwares are used as platform. The same heterogeneous situation applies to users who need to access the VO, because users holding one kind of credential could need to travel through multiple resources with different security approaches. To solve the above issues, the Grid middlewares should provide the following ability:

- The user which participates the VO should be able to access data and resources without being required to constantly do authentication at each resource or data;
- The VO should be able to easily manage the privileges or memberships of the users;
- The VO together with its resources should be able to easily manage the authorization, given the condition that the resources are owned by different administrative domains.

In grid community, Grid Security Infrastructure (GSI)[2] is the *de facto* for authentication and transport level communication, which builds on Public Key Infrastructure (PKI), an architecture based on X.509 public key certificate. GSI implements some enhancement (such as certificate delegation) based on standard SSL/TLS protocol. Mutual authentication is required by GSI, and is the default configuration for GSI based grid deployment. The X.509 certificates are required for both the client and service sides in order to achieve mutual authentication. The X.509 certificates are issued by trusted third parties called certificate authorities (CA), and then CAs constitute trust federation and guarantee two different X.509 certificates from different CAs can accomplish authentication to each other. So in the GSI based Grid system, if a user would access Grid system, he/she should own a X.509 certificate which is issued by a CA that is trusted by other entities.

In terms of the virtual organization issues mentioned earlier, GSI and some other related solutions like Virtual Organization Management Service (VOMS) [3] have partly solved them in some sense. The identity delegation and X.509 proxy certificate initiated by GSI can achieve single sign-on, so that the need to re-enter the user's pass phrase for authentication can be avoided by creating the proxy certificate. As the most successful Grid authorization model, VOMS adopts Attribute Certificate[4] approach. Attribute Certificate is signed by VOMS server and then binded to proxy certificates by client, and attributes certificate includes the VO memberships which is associated with the proxy certificate's identity. Eventually the resources can make access control decision based on parsing the VO membership attributes from the Attribute Certificate.

The VO is essentially supposed to be able to provide resource sharing capability with resources running under heterogeneous systems/middlewares. This goal can be achieved using Web Service Architecture (WSA). Therefore, the Web Service technology has been converged into Grid computing middlewares. The Web Service technology has been adopted by Globus toolkit[5], gLite[6], as well as Advanced Resource Connector(ARC) [7]. In fact, utilizing Web Service technology and providing seamless interoperability with other middlewares is one of the main goals of the new version of ARC middleware.

Unfortunately, the existing grid security solutions such as GSI and VOMS do not apply to Web Service based applications very well due to the following reasons:

- The GSS API based confidential communication has not been adopted by Web Service implementations.
- The Attribute Certificate has not been adopted by any Web Service implementation at all.
- Web Service implementations commonly use the Web Services Security(WS-Security)[8], Security Assertion Markup Languages (SAML)[9], and standard SSL/TLS as well as standard X.509 certificate when considering security, which make them impossible to interoperate with the GSI based Grid middlewares.
- Finally, identity delegation is a good solution for user or user's delegatee (on behalf of the user) to move/travel from one resource to another resource. But since the identity delegation has been completely coupled with the GSI confidential communication process, it is not possible for the existing identity delegation solution to be used by Web Service applications.

Moreover, the grid security solutions require each user to possess a X.509 certificate. But the process to get a certificate may take quite long time, since the requester needs to be checked and approved by the Registration Authority (RA) and then the Certificate Authority (CA) can issue the certificate according to the approval from RA. Meanwhile the process to use the certificate is also not so easy for the wider less-IT focused research community to deal with, since the user need to generate the proxy certificate by using the grid specific command lines, such as *grid-proxy-init*, and *voms-proxy-init*. On the other hand, users could often have some local community/institutional credentials such as username/passwords to which users are more familiar. Enabling users to use their own community credential instead of the X.509 certificate to access Grids is a promising solution to make the Grids more easily accessible.

The above analysis shows that in order to achieve interoperability and accessibility for user to access Grid system which is established on different middlewares, we need to address the mismatch between the different security solutions from these middlewares in order to ensure the interoperability as wide as possible; it is also necessary to address the mismatch between users' usual experience and middleware's requirements in order to ensure the access as simple as possible.

The approach in the new version of ARC is to utilize the existing Web Service standards, such as SAML and WS-Security, to achieve the interoperability and accessibility without breaking existing Grid specific security protocols.

The rest of this paper is organized as follows: Section 2 presents the ARC Grid middleware, especially the architecture of the new version of ARC. Section 3 describes the solution for cross-middleware authentication and single sign-on, including implementations for bridging the mismatch in terms of authentication and single sign-on. Section 4 discusses the related research, and Section 5 presents conclusion and future work.

II. ARC Grid Middleware

ARC(Advanced Resource Connector) is an open source Grid middleware solution released under Apache license. ARC middleware initially aimed at developing self-organized, fault-tolerant, non-intrusive, easy-manageable Grid middleware [10]. Now the classic version of ARC provides Grid services for job submission and management, resource characterization, resource aggregation and discovery, basic data management, integration of grid security solution, etc. Classic ARC has been deployed and used in production environment, and been one of the widely deployed Grid middlewares in Europe.

The new generation of ARC is developed by KnowARC project [11], based on the functionality and capabilities of the classic ARC middleware. It aims at implementing a Web Service oriented Grid middleware which will provide higher levels of resource and user abstraction through well-defined Web Service interface [12] in order to provide interoperability with other service-based Grid middlewares, as well as other Web Service compatible applications.

As the key part of the implementation of new ARC middleware, there is a lightweight Web Service container called Hosting Environment Daemon (HED) which provides a hosting for various services at application level, as well as a bunch of modules to support flexible, interoperatible, and efficient communication mechanism for building SOAP based Web Services. The whole design of the HED is built around the idea of flexibility and modularity, which means for the developer, he can easily concentrate on the application level Web Service implementation by only using the core minimum amount of components if he only would do application level development, or he can work on the middleware level implementing another communication protocol or authentication mechanism by using the core minimum amount of components and external dependencies; also for the deployer, he can easily configure and deploy the middleware and application for different kinds of requirements without being bothered to know much about the implementation.

The architecture of the HED is illustrated by Figure 1. In general, there are few components called Message Chain Component (MCC) which are in charge of implementing different protocol levels. For instance, as shown in the example message flow, HTTP MCC will process stream from TLS MCC to parse HTTP message and pass its body to SOAP MCC, and also process SOAP response from SOAP MCC to generate HTTP message for TLS MCC.

Dotted lines at Figure 1 show alternative path information can propagate among MCCs. Service administrator can configure the MCCs according to the interoperability requirements with the other part. For instance, the configuration marked with dashed lines is compatible to WSE (Web Services Enhancement for .NET) 's SOAP message mechanism (see WSE's *SoapSender* and *SoapReceiver*). Another configuration could be SOAP over HTTPG (HTTP over GSI) which is needed to interoperate with services like Storage Resource Manager (SRM)[13] service. This shows the flexibility of HED in terms of protocols support.

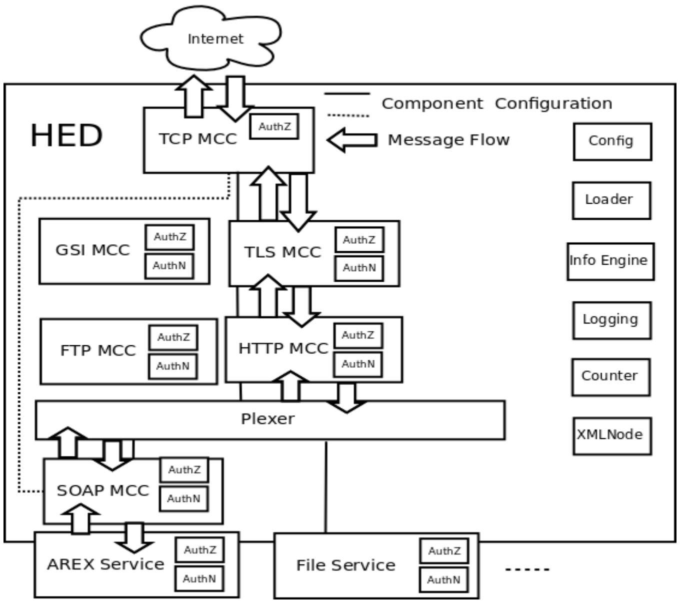

Figure 1. The example of Host Environment Daemon deployed with A-REX and File services

HED contains a framework for implementing and enforcing authentication and authorization. Each MCC or Service has a common interface for implementing various authentication and authorization functionality. Each functionality can be implemented as a pluggable and configurable component (plug-ins) called *SecHandler* which is C++ class and provides method for processing message that travels through MCC or Service. Each MCC or Service is usually configured with two queues of *SecHandler* – one for incoming message and one for outgoing message respectively. All *SecHandler* components attached to the queue are executed sequentially. If any of them fails, message processing fails as well. In Figure 1, the "*AuthZ*" and "*AuthN*" sub-modules inside MCCs and Services are examples of *SecHandler*.

As explained above, HED is supposed to act as a Web/Grid Service container, but on the client side, from implementation perspective, there is similar architecture implemented for processing messages from different protocols and handling security functionality, also there is a specific application programming interface (API) for developers to easily write Web Service client code.

III. CROSS-MIDDLEWARE AUTHENTICATION AND SINGLE SIGN-ON

We exploited the authentication and single sign-on issue in the scenario that user or user's delegatee (act on behalf of this user) needs to access across different middlewares. Our work comprises the following five major aspects:

- *Integration of community authentication* --- to utilize the standard-compatible community authentication mechanism from Shibboleth [14] Identity Provider software when users authenticate against Grid services.
- *Short lived credential service* --- to issue the short lived X.509 certificates for users by using the community authentication, in order to interoperate with services that depend on PKI based mutual authentication.
- *Credential delegation service* --- to process the X.509 credential delegation as a normal Web Service, in order to provide single sign-on ability.
- *Message level authentication* --- to provide message level authentication by implementing WS-Security.
- *SAML token delegation service* --- to process the delegation of WS Security SAML token in order to provide single sign-on for SOAP message level authentication.

A. *Community based authentication for Grid by utilizing Shibboleth Identity Provider.*

AAI (Authentication and authorization Infrastructure) is a solution for the authentication and authorization for inter-organization resource sharing, such as electronic resource sharing between libraries, etc. AAI implicitly applies to community or institutional based authentication where users are from different home communities but need to get resources from other communities by using some federation mechanism. Unlike the X.509 based authentication solution in current grid systems, AAI does not require user to provide X.509 certificate, instead, it can support different types of authentication, such as username/password authentication, IP address authentication, etc. There are a few implementations about AAI, among which Shibboleth is one implementation which has been widely deployed.

Shibboleth provides cross-domain single sign-on and attribute-based authorization while preserving user privacy. It is based on the OASIS Security Assertion Markup Language (SAML), especially the new version of Shibboleth supports SAML 2.0 specification. For authentication, the main SAML profile that Shibboleth implements is the SAML2.0 web browser SSO profile, which define two functional components, an Identity Provider and a Service Provider. The Identity Provider (IdP) is responsible for creating, maintaining, and managing user identity, while the Service Provider (SP) is responsible for controlling access to services and resources by using the SAML assertion produced and issued by IdP upon request. In order to discover which home community does a user come from, Shibboleth specifies an optional third component called "Where Are You From?" (WAYF) service to aid in the process of IdP discovery, and this IdP discovery process is also standardized and defined in SAML 2.0 specification and called as "Identity Profile Discovery Profile"

We utilized the SAML2.0 web browser SSO profile for the authentication in ARC middleware. But since the SSO profile is initially supposed to protect web applications and provide authentication for web users, we implemented some external code on the client and service side to integrate SSO profile. On the client side, apart from the client interface for writing Web Service client, we implemented the user agent functionality of web browser in order to mimic its behavior, such as http redirection, cookie processing (In fact, the implementation of user agent is also based on the client interface of ARC, specifically, the https client interface, since the client interface of ARC can support different protocols which are incarnated by different MCC). Hence client developers who would use SAML2.0 SSO profile should call the user agent interface and then the Web Service client interface. On the service side, we implemented the Service Provider functionality (based on the HTTP MCC configured together with TLS MCC) which is called SP Service. For Identity Provider, the Shibboleth IdP implementation is used. Figure 2 shows the process of SAML2.0 SSO integrated in ARC client and service.

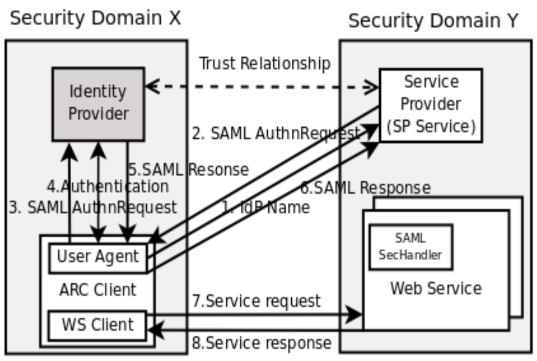

Figure 2. SAML2.0 SSO profile in ARC

The steps shows in Figure 2 are described as follows:

1. The client uses the user agent interface to launches a http request including the IdP name (to which the user belongs) to the service side. The endpoint of the SP service is the same as that of the other target services, except the last part of the endpoint is “saml2sp” which is specific for pointing to the SP Service. Note that we use Identity Provider (IdP) name here to simplify the IdP discovery process in order to avoid the IdP discovery process, because we suppose that user who would access the target services should better know where is he from initially.

2. The SP Service (Service Provider) searches the metadata (we use the same metadata format defined in Shibboleth) and gets the location of the single sign-on service (hosted in IdP) and also the location of assertion consuming service (hosted in this SP itself) in order to compose the SAML <*samlp:AuthnRequest*> message. Then SP Service issues this <*samlp:AuthnRequest*> message by using its own X.509 certificate (Note in the SAML SSO profile, X.509 certificates are still needed for IdP and SP) and sends back to user agent.

3. User agent sends the <*samlp:AuthnRequest*> message to Identity Provider.

4. Identity Provider requires an act of authentication. The authentication mechanism is outside of the SAML2.0 SSO profile. Shibboleth IdP implementation chooses some login handlers for authentication. The current user agent implementation is compatible to *Username/Password* login handler of Shibboleth IdP. Through the http protocol, user agent will feed IdP with the username/password which has been given by the caller of user agent interface.

5. Once the authentication has been succeeded, the IdP issues a SAML response including an encrypted (encrypted by destination SP's public key) SAML assertion, and then this SAML response will be delivered by the user agent to the Service Provider.

6. The SP Service verifies and checks the SAML response, decrypts and stores the SAML assertion into session/connection context. The SAML assertion includes the <*saml:AuthnStatement*> and <*saml:AttributeStatement*>.

7. The WS client launches the Grid/Web Service request via the same connection as the one which is used by user agent to contact SP Service.

8. The Grid/Web Service checks the <*saml:AuthnStatement*> from the session context to see if the session is still valid through the *SecHandler* called “*SAML SecHandler*”. If valid, service handles the service processing and returns the response to WS client. Note that service requires that WS client is from the same connection as the one on which user agent contact SP service, in order to guarantee that the validity of SSO profile result effects the WS client/Web Service interaction.

The SP service and other functional service(s) are hosted by the same container, and they use the same X.509 credential. The client authentication is switched off, so that client doesn't need to use any X.509 credential. Only the trusted certificates (CA certificates for both SP and IdP) need to be configured for client side so that SP and IdP can authenticate themselves to the client. As required by SAML2.0 profile, the SP and IdP should have trust relationship to each other.

One benefit of SAML2.0 SSO profile that is worth mentioning is: the Identity Provider could cache the authentication result through session management once the user agent has succeeded to authenticate, then in a short period this authentication result is valid so that user agent doesn't need to feed IdP with user's username and password the next time (if this point of time is not out of the scope of valid period) it authenticates against IdP. So user (or the client on behalf of this user) can travel across multiple security domains with only inputing his name and password once, which is the characteristic of single sign-on.

Since the Shibboleth implementation of SAML is standard-compliant and widely deployed, the solution implemented in ARC can easily interoperate with other SAML implementations with minimum change, and more importantly, this solution can succeed to utilize the widely deployed SAML implementation for authentication in grid systems by avoiding the usage of X.509 certificate.

Moreover, even though the implementation is based on ARC middlerware, the idea can be adopted by other Grid middlewares if they only require server authentication instead of mutual authentication.

B. Short-lived credential service

However, most of the widely used grid middlewares are based on GSI while GSI requires mutual authentication. Also for Web Service based grid solution, we can not prevent service side from requiring client X.509 certificate. Based on the solution described in Section 3.1, we implemented a short lived credential service (SLCS) by which user can get a short-lived X.509 certificate without being bothered to contact any registration authority (RA) or certificate authority (CA).

The SLCS service is also a Web Service (standard web service implemented by using ARC service interface), and the SLCS client is a specific command-line interface(CLI) which includes the user agent and WS client. The whole process of SLCS invocation is showed in Figure3 (from step 1 to step 8), which is the same as in Figure 2, except that step7 and step8 is incarnated for SLCS certificate request and response.

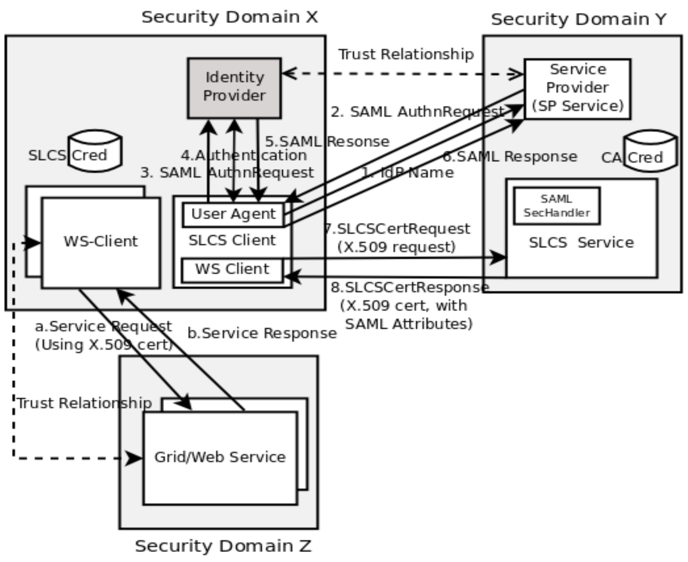

Figure 3. Short lived credential service

SLCS client generates a X.509 certificate request, launches a Web Service request which includes the certificate request; SLCS service then gets the certificate request, composes a distinguished name (DN), issues a certificate (short lived, 12 hours by default) with the SAML attribute (from the SAML2.0 SSO profile) as the X.509 certificate extension, and puts the certificate in to the Web Service response; SLCS client get the response and stores the X.509 certificate into local repository.

The CLI for the SLCS client is like this:

```
$ ./arcslcs -S https://127.0.0.1:60000/slcs -I https://idp.testshib.org/idp/shibboleth -U myself -P myself
```

Since the lifetime of the short lived credential is normally short, it is not a must to protect the private key by pass phrase. As illustrated in step a and step b in Figure 3, if the private key is not protected, through Web Service client, the user can use the X.509 certificate to access Grid Service or Web Service from any kind of middleware. If the private key is protected, he can use the X.509 certificate to generate a proxy certificate (by using command-line interface utility such as *grid-proxy-init*, *voms-proxy-init*), and then use the proxy certificate to access Grid/Web Service.

It is worth mentioning that since ARC middleware can support GSI communication by configuring the GSI MCC, together with the X.509 certificate, the Web Service client developed by ARC Web Service client interface can interoperate with Grid Service that requires GSI communication.

It might be noticed that how to compose the distinguished name (DN) for the certificate is a critical issue for SLCS service. Since the Shibboleth Identity Provider uses the eduPerson schema [15] for the definition of <*saml:Attribute*> in <*saml:AttributeStatement*>, we pick the relatively distinguishable attribute "*eduPersonPrincipalName*" for the DN. A typical *eduPersonPrincipalName* value could be *alice@example.org*, then the DN is "*/O=knowarc/OU=example.org/CN=alice*".

The obvious benefit of SLCS service is that: If a user passes the authentication to his home Identity Provider, he can get the X.509 credential at anywhere only by running the SLCS client command together with inputing his username and password to this home IdP, and then access the grid system conveniently; meanwhile, thanks to the single sign-on characteristic of SAML2.0 SSO profile, this user doesn't not need to input his username and password in a valid period after the first time he succeeds to authenticate against his home IdP by running the SLCS client command on the same node, even if this SLCS client command is supposed to run against a few SLCS services to get a few SLCS credentials.

C. X.509 Credential delegation service

The delegation of credentials proposed in Grid Security Infrastructure (GSI) is a good solution for the supporting of single sign-on for users, except the coupling of the GSI communication which is not so widely accepted in Web Service applications. In the design of the new version ARC middleware, we keep the credential delegation idea, while utilizing the SOAP communication (together with TCP or TLS for transport level communication) instead of the GSI communication for the credential delegation process.

As shown in Figure 4, there is a specific WS client and Web Service for processing delegation: delegation client, and delegation service. The *WSDL*(Web Service Description Language) of delegation service has two main operations, one for processing the delegation initiation, and the other for storing the signed proxy certificate. The delegation protocol is detailed in Figure 5.

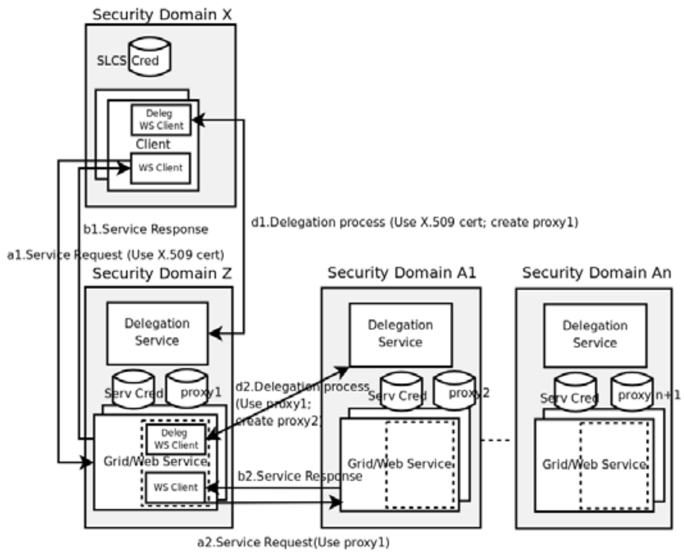

Figure 4. Credential delegation service and client

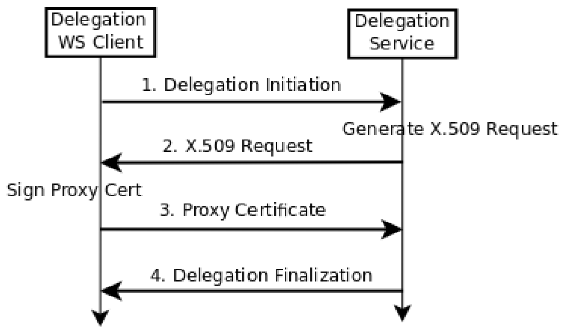

Figure 5. Credential delegation protocol

For the delegation process, the delegation WS client functionality (inside each delegation service or the initial client) which is included to access the target delegation service (d1 or d2 in Figure 4), should use the user's X.509 credential or delegated credential (on behalf of the user) for authentication and secure communication, as well as delegating one more level of delegated credential. For other general service invocation process, the WS client functionality (inside each service or the initial client) which is included to access the target service (step a1 or a2 in Figure 4), should also use the user's X.509 credential or delegated credential for authentication and secure communication.

Since each WS client accesses the target service (delegation service or other general services) on behalf of the user, the trust relationship between the user's certificates (X.509 certificate and proxy certificate) and services is required, while the trust relationship between service's certificate and another service's certificate is not required.

What should be mentioned is that the delegation service is also a normal Web Service. Therefore, delegation service implemented in ARC can be accessed by any standard-compliant WS client that intends to delegate credential; also delegation client implemented in ARC can access any other standard-compliant Web Service which accepts delegation request.

D. Message level authentication

Transport level security (TLS or SSL) is not sufficient for securing Web Service, because the secure channel created by SSL/TLS make it impossible to differentially protect the SOAP message, e.g. to encrypt or sign only particular components of the SOAP message, which is relevant when non-sensitive portions of the message need to be accessed or changed by intermediate actors. Additionally, SSL/TLS is incapable of providing end-to-end protection for SOAP message which might flow through multiple intermediate actors, because it only allows each hop to be protected with the resultant security gaps at intermediate actors. Web Services Security (WS-Security)[8] specification provides means for applying security to Web Service.

WS-Security related specifications are implemented as *SecHandler* in ARC. There are three kinds of WS-Security version 1.1 specifications which are completely or partly implemented: Username Token profile, X.509 Token profile, and SAML Token profile.

For those WS clients on behalf of a user, at any step of service invocation, the WS client can augment the SOAP message with WS-Security token by simply configuring the WS-Security *SecHandler* into container's configuration, if the relying party (target service) requires.

If the WS client owns an X.509 certificate or proxy certificate, the process of generating an X.509 Token for a SOAP message is: this client puts the content of certificate into X.509 token as *<wsse:BinarySecurityToken>*, and uses the private key (corresponding to the certificate) to create XML signature for the SOAP message body.

However the process of generating a SAML Token is more complicated. In WS-Security SAML Token profile, *ConfirmationMethod* is the verifying method for establishing proof-of-possession for the subject. Two methods are proposed in SAML Token profile: *holder-of-key*, *sender-vouches*. We rely on the *holder-of-key* method. For the *hold-of-key* subject confirmation method, there are three interaction parts: the *attesting entity*, the *relying party* and the *issuing authority*. Firstly the attesting entity authenticates to issuing authority by using some authentication scheme such as X.509 Token profile, or SSL/TLS mutual authentication. So then issuing authority is able to make a definitive statement (sign a SAML assertion) about an act of authentication that has already taken place. The attesting entity gets the SAML assertion and then signs the SOAP message together with the assertion by using its private key (the relevant certificate has been authenticated by issuing authority, and its relevant public key has been put into *SubjectConfirmation* element under SAML assertion by issuing authority). Only the exact entity that possesses the private key which is paired with the public key in the SAML assertion can sign the SOAP message, which establishes an irrefutable connection between the author of the SOAP message and the assertion describing an authentication event.

The relying party is supposed to trust the issuing authority. When it receives a message from the asserting entity, it will check the SAML assertion based on its predetermined trust relationship with the SAML issuing authority, and check the signature of the soap message based on the public key in the SAML assertion without directly trust relationship with attesting entity (holder of this public key).

In terms of verification, since the proxy certificates has been introduced in IETF (RFC 3820)[16], and OpenSSL has also implemented RFC3820, it is not difficult to use the proxy certificate for X.509 Token profile to issuing X.509 token.

E. SAML Token delegation service

There is also delegation requirement on the message level: some entities often need to temporarily delegate itself to another entity to perform action on their behalf. We propose a delegation mechanism for SAML Token. Figure 6 shows the participant parties of SAML Token delegation.

The delegation process is similar to the process in X.509 credential delegation service, with the following differences:

1. Transport level communication between delegation client and service is not based on mutual authentication between proxy certificate and service certificate; instead, it is based on mutual authentication between service certificate and service certificate.
2. Delegation service will not generate any certificate request, instead, the delegation client will pick the public key from peer certificate after successful SSL/TLS authentication. So step2 of the delegation protocol is only some confirmation information which informs delegation client to proceed.
3. For those multiple intermediate services through which a SOAP message flows, delegation service does not need to appear, instead, it only need to appear on the end of the SOAP message flow.
4. SAML Token might include one or a few <*saml:Attribute*> which carries the attribute information related to the subject, instead of the Attribute Certificate which is used in VOMS proxy certificate for carry the attribute information

Figure 7 shows the SAML Token (SAML Assertion) inside SOAP message is changed due to delegation (we suppose X.509 token is used for attesting entity to authenticate against issuing authority). The SOAP message on the left side is the one which B sends to relying party (intermediate service). B is the identity of the attesting entity's X.509 token, while A is the identity of issuing authority, and the SOAP message is signed by B's private key which is corresponding to the X.509 certificate that is used to create the X.509 token. On the right side, C is the identity of the delegation service (also C is the identity of the target service to which B will access), and now the SOAP message is signed by C's private key which is corresponding to the target service's certificate.

What is worth mentioning is: since the SLCS X.509 credential or delegated credential can be used to issue X.509 token, the X.509 credential delegation and SAML Token delegation are smoothly bridged, which benefits the single sign-on.

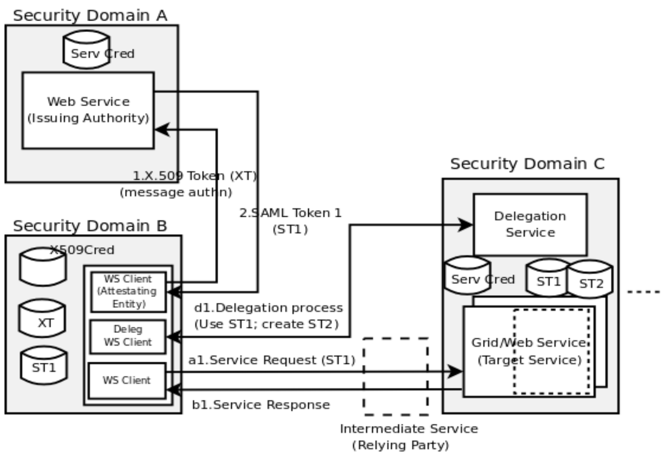
Security Domain A
Serv Cred
Web Service (Issuing Authority)
1.X.509 Token (XT) (message authn)
2.SAML Token 1 (ST1)
Security Domain B
X509Cred
XT
ST1
WS Client (Attestating Entity)
Deleg WS Client
WS Client
d1.Delegation process (Use ST1; create ST2)
a1.Service Request (ST1)
b1.Service Response
Intermediate Service (Relying Party)
Security Domain C
Delegation Service
Serv Cred
ST1
ST2
Grid/Web Service (Target Service)

Figure 6. SAML Token delegation service and client

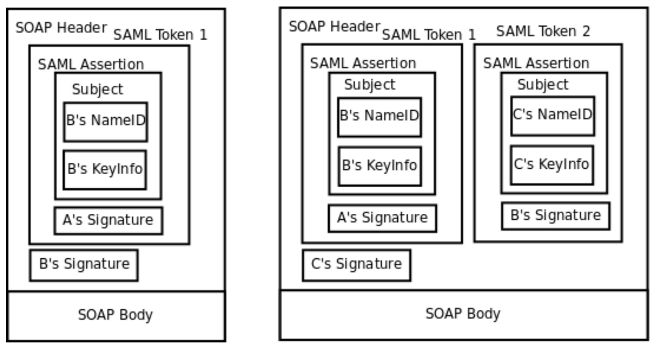

Figure 7. SOAP message header with SAML delegation token inside

IV. RELATED WORK

Similar work that integrates Shibboleth and Grid middleware does exist. GridShib[17][18] uses Shibboleth to get entity's attribute for the authorization of services developed in Globus Tookit. Unlike the work of this paper, GridShib does not use Shibboleth's native login handlers for authentication to Shibboleth IdP, instead, a new login handler specific for X.509 based authentication is developed for Shibboleth IdP, and a credential service is developed for acting as an online-CA, authenticating the user, and issuing short-lived X.509 credential. Moreover, this paper provides a single sign-on solution for user with community credential authenticating to Grid services by directly using SAML2.0 SSO profile.

SWITCH short-lived credential service (SLCS) [19] provides a SLCS service and client. The benefit of our work comparing to the SWTCH's solution is the SLCS service and client is based on Web Service, so that it can be implemented in any Web Service container while keeping interoperability.

In terms of delegation, apart from the credential delegation mechanism from GSI [2][20], [21] proposes a delegation protocol based on the WS-Trust [22] in order to provide a standard and interoperable protocol for delegation in Grids. WS-Trust will also be adopted by ARC middleware to express the specifications required to define delegation protocol in a standard way. Gridsite project implements a X.509 credential delegation solution [23] based on Web Service, with which ARC delegation client can easily interoperate.

GSI plug-in for gSOAP[24] adds GSI support for gSOAP Web Service tookit. The GSI MCC implemented in ARC can also achieve the same functionality as this plug-in. Moreover, the ARC solution is more flexible than it, because GSI MCC of both service and client sides are pluggable, which means service and client developer of ARC do not need to modify the service or client implementation in order to add or replace the GSI functionality.

V. Conclusion and Future work

Within this paper we presented the solution inside ARC Grid middleware towards the cross-middleware authentication and single sing-on. The goal of our work is to develop an interoperatable and accessible single sign-on solution to be used for VO scenario in production Grids, where users can easily authenticate against the Grids with any of his credentials (not only X.509 credential) from anywhere, and travel inside Grid middleware boundaries and even across heterogeneous Grids middleware boundaries, without re-authentication in a reasonable period. In order to do so, we implemented some standard based Web Services (including SLCS service, X.509 credential delegation service, and SAML Token delegation service), as well as the user agent and Service Provider in context of SAML2 SSO profile.

It must be noted that due to standard specifications utilized (SAML and WS-Security) which are independent from any particular middleware, though the WS clients and Web/Grid Services presented in this paper are mostly developed on ARC middleware, they can interoperate with WS clients and Web/Grid Services developed on any other middlware in a standardized way. ARC also supports interoperability with Grid middlewares that require GSI based communication.

Though only authentication issue is discussed in this paper, as the future work, we will implement a solution for cross-middleware attribute-based authorization by using the standard specification such as SAML.

Acknowledgment

We would thank all of the ARC middleware developers for their intelligent work. The work has been supported by the European Commission through the KnowARC project (contract Nr. 032691).

References

[1] Foster I, Kesselman C, Tuecke S. The anatomy of the Grid: Enabling scalable virtual organizations. International Journal of Supercomputer Applications 2001; 15(3):200–222.

[2] Foster, I., Kesselman, C., Tsudik, G. and Tuecke, S. A Security Architecture for Computational Grids. ACM Conference on Computers and Security, 1998, 83-91.

[3] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, K. Lorentey, and F. Spataro. From gridmap- file to voms: managing authorization in a grid environment. Future Generation Comp. Syst., 21(4):549–558, 2005.

[4] RFC 3821- An Internet Attribute Certificate Profile for Authorization. http://www.faqs.org/rfcs/rfc3281.html

[5] Globus Toolkit. http://www.globus.org/toolkit/

[6] gLite: lightweight middleware for grid computing. Http://glite.web.cern.ch

[7] *Advanced Resource Connector.* http://www.nordugrid.org/middleware/

[8] *OASIS Web Services Security.* www.oasis-open.org/committees/wss/

[9] *OASIS Security Assertion Markup Languages (SAML).* www.oasis-open.org/committees/security/

[10] *M. Ellert, M. Grønager, A. Konstantinov, B. Konya,* J. Lindemann, I. Livenson, J. Langgaard Nielsen, M. Niinimaki, O. Smirnova, and A. Waananen. Advanced resource connector middleware for lightweight computational grids. Future Generation computer systems, 23(2):219–240, 2007.

[11] KnowARC project. https://www.knowarc.eu/

[12] *Design document of new version ARC.* https://www.knowarc.eu/documents/Knowarc_D1.1-1_07.pdf

[13] *A. Shoshani, A. Sim, and J. Gu,* Storage Resource Managers: Essential Components for the Grid, *Grid Resource Management: State of the Art and Future Trends, Kluwer Publishing, 2003.*

[14] The Shibboleth Project. Http://shibboleth.internet2.edu/

[15] eduPerson and eduOrg Object shema. http://middleware.internet2.edu/eduperson/

[16] *RFC 3820 - Internet X.509 Public Key Infrastructure (PKI)* Proxy Certificate Profile. http://www.ietf.org/rfc/rfc3820.txt

[17] *V. Welch, T. Barton, K. Keahey and F. Siebenlist. Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration. 4th Annual PKI R&D Workshop, 2005.*

[18] *T. Barton, J. Basney, T. Freeman, T. Scavo, F. Siebenlist, V. Welch, R. Ananthakrishnan, B. Baker, and K. Keahey. Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, Gridshib, and MyProxy. 5th Annual PKI R&D Workshop, 2006.*

[19] *SWITCH Short Lived Credential Service. http://www.switch.ch/grid/slcs/*

[20] *V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Teucke, J. Gawor, S. Meder and F. Siebenlist. X.509 proxy certificate for dynamic delegation, Proceeding of the 3rd Annual PKI R&D Workshop, 2004.*

[21] *M. Ahsant, J. Basney, and O. Mulmo. Grid Delegation Protocol, UK Workshop on Grid Security Experiences, Oxford. 2004.*

[22] *OASIS WS-Trust specification. http://docs.oasis-open.org/ws-sx/ws-trust/200512*

[23] *gridsite delegation service. http://www.gridsite.org/wiki/Delegation_protocol*

[24] Giovanni Aloisio, Massimo Cafaro, Italo Epicoco, Daniele Lezzi, and Robert van Engelen, The GSI plug-in for gSOAP: Enhanced Security, Performance, and Reliability, in the ITCC conference 2005, IEEE Press, Volume I, pages 304-309.