Authentication Protocol For Cross Realm SOA-Based Business Processes

Author : Layth Hamzah Kamil

Abstract— This Modern distributed application is embedding an increasing degree of dynamism, from dynamic supply chain management, enterprise federations, and virtual collaborations to dynamic service interactions across organizations. Such dynamism leads to new security challenges. Collaborating services may belong to different security realms but often have to be engaged dynamically at run time. If their security realms do not have in place a direct cross-realm authentication relationship, it is technically difficult to enable any secure collaboration between the services. Because organizations and services can join a collaborative process in a highly dynamic and flexible way, it cannot be expected that every two of the collaborating security realms always have a direct cross-realm authentication relationship. A possible solution to this problem is to locate some intermediate realms that serve as an authentication-path between the two separate realms that are to collaborate. However, the overhead of generating an authentication-path for two distributed realms is not trivial. The process could involve a large number of extra operations for credential conversion and require a long chain of invocations to intermediate services. This problem is addressed by presenting a new cross-realm authentication protocol for dynamic service interactions, based on the notion of multi-party business sessions. This protocol requires neither credential conversion nor establishment of any authentication path between session members. The main contributions of this work are: (1) using the multi-party session concept to structure dynamic business processes, (2) a simple but effective way to establish trust relationships between the members of a business session, and (3) a set of protocols for multi-party session management.

Keywords: Authentication, Kerberos, multi-party session, SOA, Web Services.

I. INTRODUCTION

Modern distributed applications are embedding an increasing degree of dynamism, from dynamic supply chain management, enterprise federations, and virtual collaborations to dynamic service interactions across organizations. Such dynamism leads to new security challenges. Collaborating services may belong to different security realms but often have to be engaged dynamically at run time. If their security realms do not have in place a direct cross-realm authentication relationship, it is technically difficult to enable any secure collaboration between the services. Because organizations and services can join a collaborative process in a highly dynamic and flexible way, it cannot be expected that every two of the collaborating security realms always have a direct cross-realm authentication relationship. A possible solution to this problem is to locate some intermediate realms that serve as an authentication-path between the two separate realms that are to collaborate. However, the overhead of generating an authentication-path for two distributed realms is not trivial. The process could involve a large number of extra operations for credential conversion and require a long chain of invocations to intermediate services.

This problem is addressed by presenting a new cross-realm authentication protocol for dynamic service interactions, based on the notion of multi-party business sessions. This protocol requires neither credential conversion nor establishment of any authentication path between session members. The main contributions of this work are: (1) using the multi-party session concept to structure dynamic business processes, (2) a simple but effective way to establish trust relationships between the members of a business session, and (3) a set of protocols for multi-party session management.

II. LITERATURE SURVEY

The issues with the cross-realm authentication have been discussed in many papers. For example both direct cross-realm authentication and transitive cross-realm authentication are supported in Kerberos [6]. By using transitive cross-realm authentication, a principal can access the resources in a remote realm by traversing multiple intermediate realms, if there is no cross realm key shared with the remote realm. However, Kerberos assumes that the authentication mechanisms in all the federated security realms are homogeneous. In practice, the authentication mechanisms in different security realms are often heterogeneous and even non-interoperable, both in structure and function. In order to address the issue of federating such heterogeneous authentication mechanisms, credential conversion mechanisms are widely used in many existing solution. The work in

[10] presents two types of credential translator services, KCA which translates Kerberos credentials to PK credentials, and KCT which translates PK credentials to Kerberos credentials.

The problems related to federation amongst heterogeneous authentication mechanisms used by different security realms are also discussed in the Web Service federation protocol [13]. The Web Service federation protocol defines a set of credential conversion mechanisms, with which a principal in a realm can convert its credential to a credential that can be accepted in another realm within the federation. It is shown that an authentication path can be found in polynomial time if there is a centralized entity which holds all the federation information of the security realms possibly involved.

Kerberos is a network authentication protocol. Kerberos is designed to provide strong authentication for client/server applications by using secret-key cryptography. This is accomplished without relying on authentication by the host operating system, without basing trust on host addresses, without requiring physical security of all the hosts on the network, and under the assumption that packets travelling along the network can be read, modified, and inserted at will. Kerberos performs authentication under these conditions as a trusted third-party authentication service by using conventional cryptography, i.e., shared secret key. The authentication process proceeds as follows: A client sends a request to the authentication server (AS) requesting "credentials" for a given server. The AS responds with these credentials, encrypted in the client's key. The credentials consist of 1) a "ticket" for the server and 2) a temporary encryption key (often called a "session key"). The client transmits the ticket (which contains the client's identity and a copy of the session key, all encrypted in the server's key) to the server. The session key (now shared by the client and server) is used to authenticate the client, and may optionally be used to authenticate the server. It may also be used to encrypt further communication between the two parties or to exchange a separate sub-session key to be used to encrypt further communication.

A service-oriented architecture is an information technology approach or strategy in which applications make use of (perhaps more accurately, rely on) services available in a network such as the World Wide Web. Implementing a service-oriented architecture can involve developing applications that use services, making applications available as services so that other applications can use those services, or both. In software engineering, a Service-Oriented Architecture (SOA) is a set of principles and methodologies for designing and developing software in the form of interoperable services. These services are well-defined business functionalities that are built as software components (discrete pieces of code and/or data structures) that can be reused for different purposes. SOA design principles are used during the phases of systems development and integration.

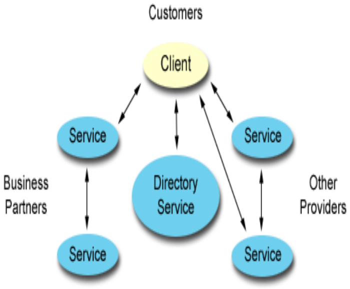


Fig 1.1

A. AUTHENTICATION ACROSS HETEROGENEOUS SECURITY REALMS

With the development of business processes, collaboration amongst organizations has become increasingly common. The emergence of web services further promotes this tendency. Such cross-organizational collaboration imposes new security challenges on authentication systems.

Because principals could join or leave a collaborative process in a highly dynamic and frequent fashion, it cannot be expected that a direct cross realm authentication relationship always exists between each pair of the collaborating security realms.

Most of existing authentication methods (e.g. Kerberos) utilizes two-party sessions to enforce their security management, which are obviously different from the method that is presented here. Generally there are two potential solutions to address the dynamic cross-realm authentication issue raised in the integration of heterogeneous security realms

1. Federated Authentication
2. Authentication path of credential conversion

B. FEDERATED AUTHENTICATION

Federated Authentication is a solution of accepting user authentication between different organizations However the overhead of generating federated authentication between many heterogeneous authentications is very high, including high infrastructure, cost of contact amendments and cost of building understandings by the partners, which leads to the slow adaptation of existing federated authentication mechanisms. Moreover, in the federated authentication, a server often needs to authenticate a chain of credentials submitted by the client and this requires the server to perform multiple extensive digital signature verifications [4]

C. AUTHENTICATION PATH OF CREDENTIAL CONVERSION

Locate some intermediate path that serve as authentication path of credential conversion between the two separate realms that are to collaborate. Generating an authentication path requires cooperation from intermediate security realms. This process requires large no of extra operations for credential conversion as well as long chain of invocations to intermediate services.

III. SYSTEM REQUIREMENT SPECIFICATION

This Software Requirements Specification provides a complete description of all the functions and specifications of the project “DYNAMIC AUTHENTICATION FOR CROSS REALM SOA-BASED BUSINESS PROCESSES”

A. Purpose

The purpose of this project is to create a solution for dynamically authenticating the services from different realms. The main contributions of this work are: (1) using the multi-party session concept to structure dynamic business processes, (2) a simple but effective way to establish trust relationships between the members of a business session, and (3) a set of protocols for multi-party session management.

B. Intended Audience

Developers, System Administration
Testers, Analysts and , Documentation writers.

C. Project Scope

The scope of this project is to design and implement a new cross-realm authentication protocol for dynamic service interaction, based on the notion of service-oriented multi party business sessions. This protocol requires neither credential conversion nor establishment of any authentication path between the participating services in business session

D. References

1. “Session Authentication protocol for web services” by S.Hada and Maruyana
2. “Dynamic authentication for multi-party service interactions” by D.Zhang
3. Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services (The Prentice Hall Service-Oriented Computing Series from Thomas Erl)
4. M.Honda,N. Nadalin, ”Securing web services” IBM systems, 2002

E. Overall Description

The remainder of this system requirement specification section is in two sub sections, the first providing a full description of the project. It lists all the functions performed by the system. The final section concerns details of each of the system functions and actions in full for the software developers’ assistance. These two sections are cross-referenced by topic; to increase understanding by both groups involved.

F. System Perspective

If a service-oriented architecture is to be effective, we need a clear understanding of the term service. A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services.

Loosely coupled .NET web services are created to demonstrate Service Oriented Architecture (SOA). I took the same example of producer, consumer and shipper from paper to implement these services respectively.

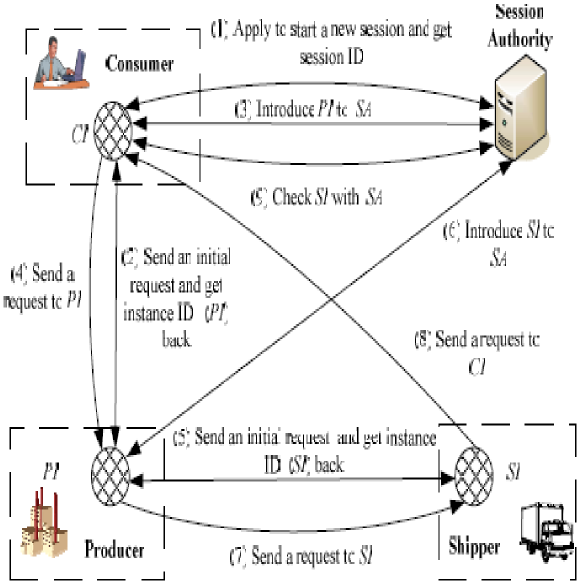


Fig 3.1.1propoed system

System Features

The proposed new cross-realm authentication protocol for dynamic service interaction has following features:

Secure multi party service interaction in services with cross-realm authentication.

Reliable and scalable solution – Any no. of session partners can be added or removed from existing session.

No overhead of credential conversion and establishment of authentication paths between collaborative session partners.

Cost effective solution as compared to existing solutions

No high infrastructure required.

Operating Environment

Operating System - Windows Vista Home Premium

Software Required – Visual Studio 2008 with .NET Framework 3.5, IIS 7.0

User Documentation

User Manual

Tutorial –

Design and Implementation Constraints

Assumptions and Dependencies

All services will adhere to Diffie-Hellman public key algorithm for generating secret keys..NET framework 3.5 is available on system where this SA is setup.

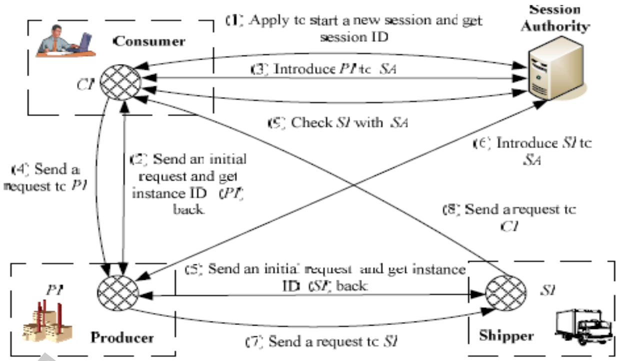
System Description

Following are features provided by proposed solution:

- MESSAGE ROUTING
- SECRET KEYS
- SESSION AUTHORITY
- NETWORK THREATS

IV. ANALYSIS

ANALYSIS MODELS



Protocol 1: Accepting a new session partner

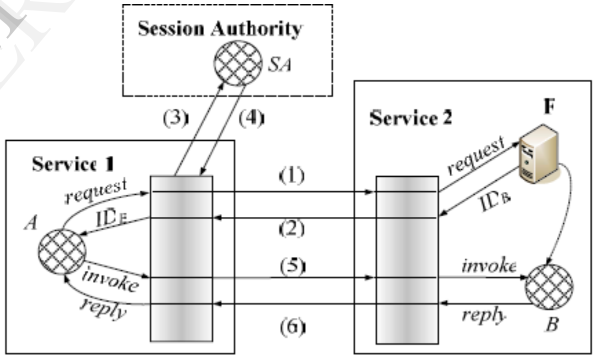


Fig 4.1Accepting a new session partner

(1) $A \rightarrow F$: $Secure(Request, ID_S, ID_A)$
(2) $F \rightarrow A$: $Secure(ID_B, ID_S)$
(3) $A \rightarrow SA$: $Valid(SP(B,S), ID_B, ID_A, ID_{SA}, ID_S, N)_{K(A, SA)}$
(4) $SA \rightarrow A$: $Valid(Confirm, N+1)_{K(SA, A)}$
(5) $A \rightarrow B$: $Valid(Invoke, ID_A, ID_B, ID_S, N_1)_{K(A, B)}$
(6) $B \rightarrow A$: $Valid(Reply, ID_B, ID_A, ID_S, N_1+1)_{K(B, A)}$
where N and N_1 are fresh nonces.

Protocol 2: Authenticating a session partner.

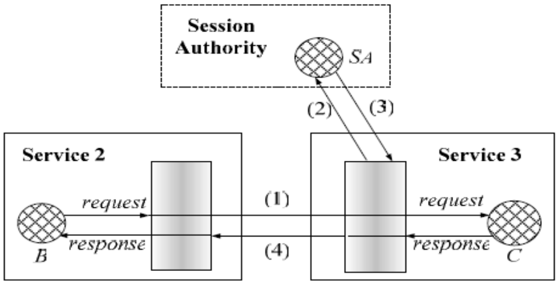


Fig 4.2 Authenticating a session partner.

(1) $B \rightarrow C$: $Valid(Request, ID_B, ID_C, ID_S, N')_{K(B,C)}$

(2) $C \rightarrow SA$: $Valid(Query, ID_B, ID_C, ID_{SA}, ID_S, N'')_{K(C,SA)}$

(3) $SA \rightarrow C$: $Valid(SP(B, S), ID_{SA}, ID_C, ID_S, N''+1\ )_{K(SA,C)}$

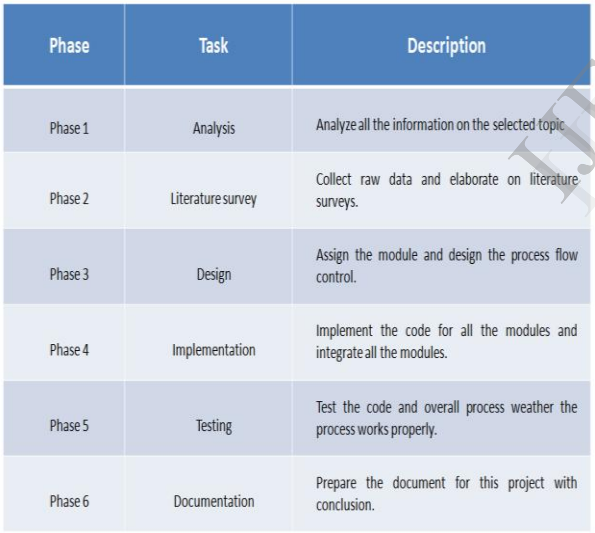
(4) $C \rightarrow B$: $Valid(Response, ID_C, ID_B, ID_S, N'+1)_{K(C,B)}$

where N' and N'' are fresh nonces.

System Implementation Plan

Phase Description:

Table II Phase Description



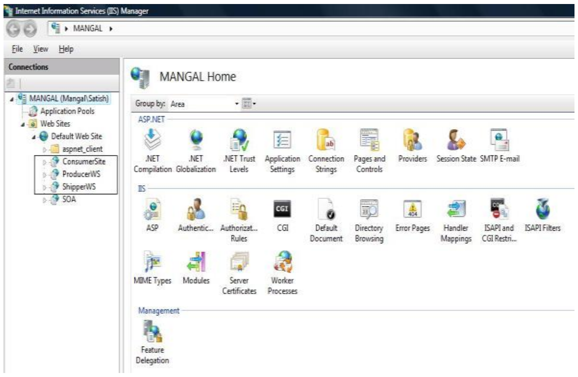
Phase	Task	Description
Phase 1	Analysis	Analyze all the information on the selected topic
Phase 2	Literature survey	Collect raw data and elaborate on literature surveys.
Phase 3	Design	Assign the module and design the process flow control.
Phase 4	Implementation	Implement the code for all the modules and integrate all the modules.
Phase 5	Testing	Test the code and overall process weather the process works properly.
Phase 6	Documentation	Prepare the document for this project with conclusion.

V. SYSTEM DESIGN

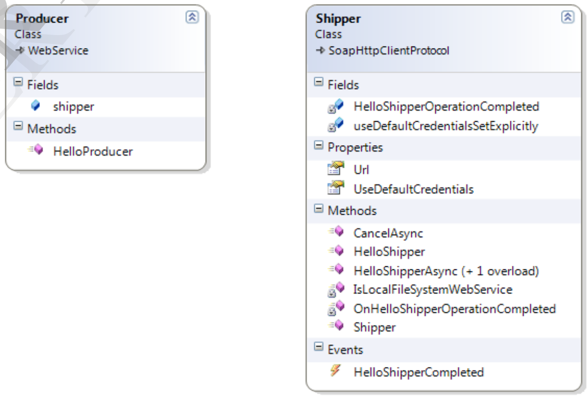
I've implemented Service Oriented Architecture (SOA). I created loosely coupled web services communicating with each other. I took the same example of producer, consumer and shipper from paper to implement these services respectively.

Implementation:

These web service are hosted in IIS as three different web application (highlighted in image below)



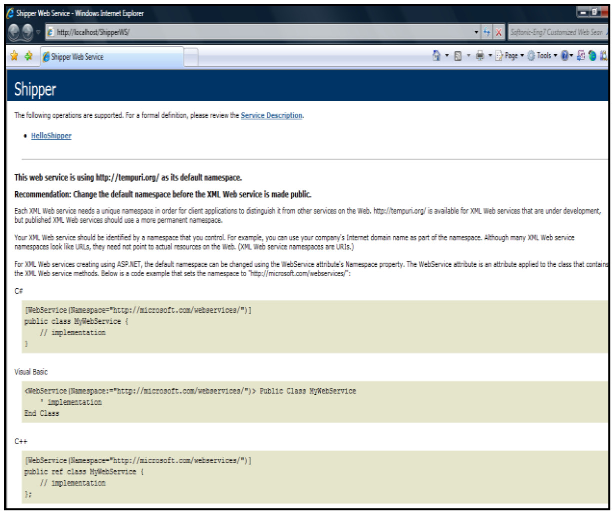
Consumer is implemented as a web site whereas Producer and Shipper are implemented as .NET web services. Consumer calls Producer web method and in turn Producer calls Shipper web method showing web service intercommunication. Web reference of IIS hosted urls are added in visual studio projects for implementation. Below figure shows Producer and Shipper web service classes.



The web services are currently anonymously accessible. However if we will implement different authentication mechanism for different services then these web service would not be communicate with each other without implementing federation/credential conversion solutions. As a next I would be implementing authentication protocol which will not only implement authentication at web service layer but also manage multi party sessions.

Output of services

Shipper



Shipper

The following operations are supported. For a formal definition, please review the Service Description.

- HelloShipper

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to "http://microsoft.com/webservices/":

C#

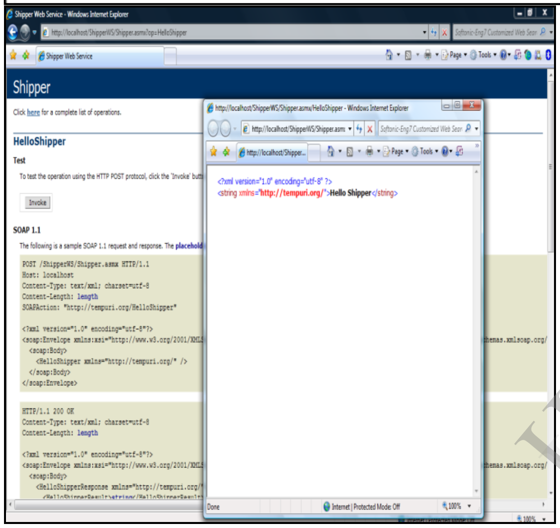
```
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
    // implementation
}
```

Visual Basic

```
<WebService(Namespace:="http://microsoft.com/webservices/")> Public Class MyWebService
    ' implementation
End Class
```

C++

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public ref class MyWebService {
    // implementation
};
```



Shipper

Click here for a complete list of operations.

HelloShipper

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' butt

Invoke

SOAP 1.1

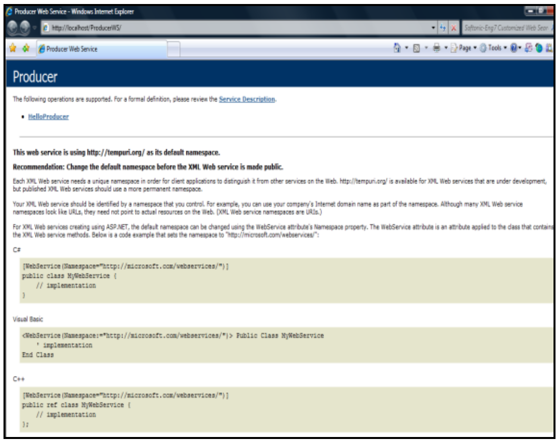
The following is a sample SOAP 1.1 request and response. The placehold

```
POST /ShipperWS/Shipper.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/HelloShipper"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XML          hemas.xmlsoap.org/
  <soap:Body>
    <HelloShipper xmlns="http://tempuri.org/" />
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XML          hemas.xmlsoap.org/
  <soap:Body>
    <HelloShipperResponse xmlns="http://tempuri.org/
      <HelloShipperResult>string</HelloShipperResult>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://tempuri.org/">Hello Shipper</string>
```

Producer



Producer

The following operations are supported. For a formal definition, please review the Service Description.

- HelloProducer

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to "http://microsoft.com/webservices/":

C#

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
    // implementation
}
```

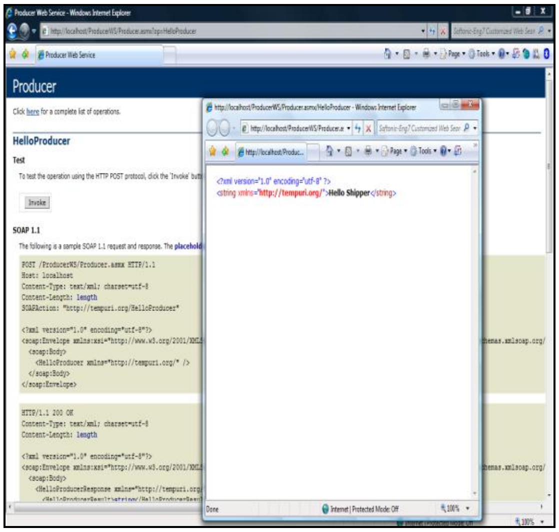
Visual Basic

```
<WebService(Namespace:="http://microsoft.com/webservices/")> Public Class MyWebService
    ' implementation
End Class
```

C++

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public ref class MyWebService {
    // implementation
};
```

Since producer webservice internally calls shipper webservice, hence executing producer from browser will give the same result which shipper webservice gives.



Producer

Click here for a complete list of operations.

HelloProducer

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' butt

Invoke

SOAP 1.1

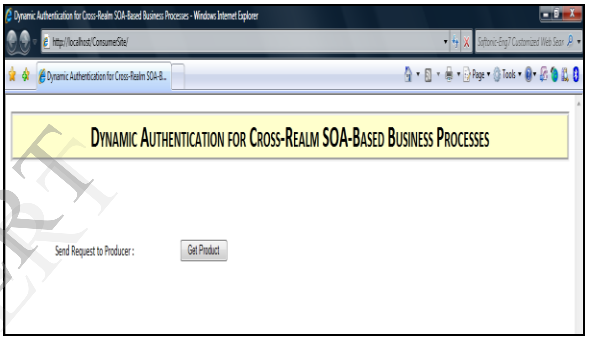
The following is a sample SOAP 1.1 request and response. The placehold

```
POST /ProducerWS/Producer.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/HelloProducer"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XML          hemas.xmlsoap.org/
  <soap:Body>
    <HelloProducer xmlns="http://tempuri.org/" />
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XML          hemas.xmlsoap.org/
  <soap:Body>
    <HelloProducerResponse xmlns="http://tempuri.org/
      <HelloProducerResult>string</HelloProducerResult>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://tempuri.org/">Hello Shipper</string>
```

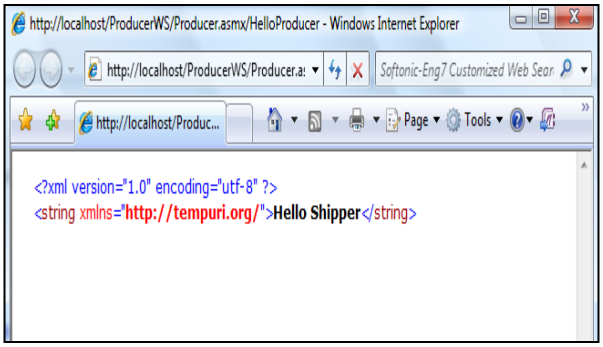
Consumer



DYNAMIC AUTHENTICATION FOR CROSS-REALM SOA-BASED BUSINESS PROCESSES

Send Request to Producer : Get Product

Clicking "Get Product" from consumer web site will call producer web service. Producer web service will internally call shipper web service to deliver product to consumer.



```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://tempuri.org/">Hello Shipper</string>
```

VI. CONCLUSION

In a web service context, a dynamic business process may involve many applications and services from different organizations and security realms, which are combined at runtime and collaborate in a peer to peer way. The dynamic authentication process between organizations could be highly complex and time consuming since intermediate authentication paths need to be created at runtime to dynamically covert credentials from different security realms. If there is no existing authentication relationship in place between two organisations, it will be practically difficult for a system to enable any secure collaboration between services from the two organizations in a just-in-time fashion.

In response to this challenge for multi-party service interactions that does not require credential conversion and establishment of authentication paths between collaborative session partners. The system also offers the ability to identify individual service instances within a business session even if some instances in fact belong to the same service. Although the amount of communications between the partners of a session and the SA is limited, the performance overhead imposed by it is indeed of some practical concern. Therefore a set of comprehensive experiments to assess the overhead on two service oriented web services are conducted.

REFERENCES

[1] S.Hada and Maruyana,"Session Authentication protocol for web services"

[2] J.D. Clercq ,"Single sign on Architectures", International conference, UK, 2002

[3] P.C. Oorschot, "Extending cryptographic Logics of belief to key agreement protocol" 1st ACM conference on computer security,1993,PP 233-243.

[4] J.Li, N. Li, X. Wang and T. Yu, "Denial of service attacks and defenses in decentralized trust management " International journal of Information security,Vol 8,2009

[5] IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 1 1 , NO. 5, JUNE 1993 "Increasing Availability and Security of an Authentication Service"

[6] W. Stallings, cryptography and network security (2nd Edition): practice Hall, Inc 1999

[7] "Distributed Access Control in CROWN Groups "Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)

[8] http://www.ibm.com/developerworks/library/ws-coor/

[9] http://www.globus.org/wsrf/specs/ws-wsrf.pdf

[10] O.Korniovaskia, P. Honeymann "kerberised credemtial Translation" A solution to web access control, USA 2001

[11] P.R. Zimmermann "The official PGP user guide "MA, USA, 1995

[12] M.K. Reiter and S.G. Stubblebine "Resilient authentication using path independence "IEEE transactions, Dec 1998

[13] M.Honda,N. Nadalin, "Securing web services" IBM systems, 2002

[14] D.Zhang, "Dynamic authentication for multi-party service interactions".

[15] OpenID specifications, 2007, http://openid.net/developers/specs/

[16] Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services (The Prentice Hall Service-Oriented Computing Series from Thomas Erl)