A Web Service Authentication Control System Based on SRP and SAML

Flávio O. Silva, M.Sc.
União Educacional Minas Gerais
Uberlândia, MG, Brasil.
flavio@uniminas.br

João A. A. Pacheco, M.Sc.
Faculdade de Computação
Universidade Federal de Uberlândia
Uberlândia, MG, Brasil.
jaapacheco@ufu.br

Pedro F. Rosa, Ph.D.
Faculdade de Computação
Universidade Federal de Uberlândia
Uberlândia, MG, Brasil.
frosi@facom.ufu.br

Abstract

Actually Internet applications can provide not only information, but also, another way of getting distributed computing. Cooperative information systems are autonomous and heterogeneous systems, distributed geographically, but interconnected. Web Services provides a set of interoperable standards that can be used to connect distributed applications. On this environment, security is a critical issue, and an attack can expose systems services without authentication. An end-to-end connection, like the ones involved in such systems, usually requires that an authentication can be shared between different information systems. Web Services security model is not yet fully defined and a lot of proposals are emerging, delaying the adoption of this technology in many situations. In this paper we present multiplatform authentication control system based on an extension of SRP protocol, using SAML. Within this solution, authentication control can be leveraged, even with weak passwords and an authentication assertion can be exchanged with different cooperative information systems.

1. Introduction

Actually Internet applications can provide not only information, but also, another way of getting distributed computing. In this scenario, the browser is not the main entry point for Internet, but virtually any application can use Internet to provide additional computing or specific services.

Cooperative information systems are autonomous and heterogeneous systems, distributed geographically, but interconnected. Internet is becoming the natural way to connect these systems. Web Services [1] provides a set of interoperable standards that can be used to connect distributed applications using the Internet.

On this environment, security is a critical issue. The wide use of Internet applications has tremendous benefits, but a security fail can expose confidential data.

On a point-to-point connection, client authentication is an effective way to protect services and information, and a common way to perform this is using a password. On an end-to-end connection, like the ones involved in cooperative information systems the security requirements are different and usually an authentication assertion has to be shared between different information systems.

Another key issue with client authentication is related with security. This approach has its weakness and an attack can break this protection and the services could be accessed without authorization.

Web Services security model is not yet fully defined and a lot of proposals are emerging, delaying the adoption of this technology in many situations.

Security is a layer in Web Services architecture and an environment for creating Web Services, such as Apache AXIS [2], does not have a way to authenticate its callers, so if the service is available, anyone can call it.

Security Assertions Markup Language (SAML) [3] provides an interoperable XML schema for exchanging authentication, authorization, and user attribute related information between different security infrastructures.

In this paper we present an authentication control system based on an extension of SRP [4] (Secure Remote Password) protocol and SAML. Using this solution, authentication control can be leveraged, even with weak passwords and an authentication assertion can be exchanged with different cooperative information systems.

The proposed solution was implemented as an independent layer. This additional independent layer

Proceedings of the IEEE International Conference on Web Services (ICWS'05)
0-7695-2409-5/05 $20.00 IEEE

can be built on any available Web Service platform. In this work we have used Apache AXIS for this end.

Section 2 introduces the Web Service main concepts. Section 3 presents Apache Axis engine; Section 4 presents the SRP protocol. Section 5 describes the authentication control proposed and its tests and results are shown in section 6. Section 7 presents some concluding remarks and potential future works.

2. Web Services Overview

A Web Service is a software application that can be remotely accessed using some protocols based on Extended Markup Language (XML) [5].

Web Services standards define the format of the message, specify the interface to which a message is sent, describe conventions for mapping the contents of the message into and out of the programs implementing the service, and define mechanisms to publish and to discover Web Services interfaces.

SOAP [6] is a XML based protocol and it is the backbone to this new generation of cross-platform, cross-language, distributed computing applications named Web Services. A SOAP message basically contain the following elements: <*soapenv:Envelope*>, <*soapenv:Body*>, and <*soapenv:Header*>.

The <*soapenv:Envelope*> element identifies the start and the end of the SOAP message. The <*soapenv:Body*> contains the message itself. Finally the optional <*soapenv:Header*> contains one or more XML elements, named header blocks.

Headers offer the mechanism by which SOAP can be extended to include additional features and functionalities to Web Service based applications, such as security, authentication, transactions, and other service level agreement attributes associated with the message.

Fault messages are generated if error occurs during SOAP messages exchange. In such case, the <*soap:Body*> element will contain only one element named <*soap:Fault*>.

Web Services protocols are based on XML standards and to get them secure XML is the natural language to perform this task.

A range of XML-based security mechanisms [7] are needed to solve all security related requirements.

Security Assertions Markup Language (SAML) [8] provides an interoperable XML schema for exchanging authentication, authorization, and user attribute related information within different security infrastructures and applications.

An assertion is a package of information that supplies one or more statements made by a SAML authority.

SAML defines three kinds of statements [8] that can be carried within an assertion. Authentication statements describes how the subject was authenticated and in what time; attribute statements provide specific details about the subject and, authorization statements identify what actions the subject is entitled to perform in a specific system.

A SAML authority is a system entity that issues assertions and a relying part is a system entity that decides to take an action based on the received assertions.

SAML defines a request/response protocol for obtaining assertions. A SAML request can either ask for a specific known assertion or make authentication, attribute, and authorization decision queries, with the SAML response providing back the requested assertions.

The language defines a binding to describe of how SAML request/responses are carried within SOAP exchange messages over Hyper Text Transfer Protocol (HTTP).

An important characteristic of SAML assertions is they can be exchanged across different enterprises, systems and platforms, thus an authentication assertion, can be sent to different systems in other to reach their available Web Services. This property is the key issue to solve the single sign-on problem.

Authentication, which is one of the security requirements, consists of an entity prove to another the identity it asserts to have. Several methods can be used to perform this operation and the username/password is the most used. In the Web Services context entities are objects that communicate with each other through SOAP messages.

SAML specification, which is based in XML assertions, predicts the use of SRP as a protocol for authentication, but until this time there is no implemented solution that uses SRP to authenticate SAML assertions.

Web Services based protocols, such as SOAP, are well defined and adopted. However, security standards are still being defined.

Although there are a lot of proposals provided by different standards organizations, at this time, there are no broadly adopted specifications for Web Services security. As a result developers can either build up services that do not use these capabilities or can develop ad-hoc solutions that may lead to interoperability problems [9].

3. Apache Axis Engine

AXIS (Apache Extensible Interaction System) essentially is a SOAP engine. It provides the infrastructure to construct applications and manipulate messages that uses SOAP.

AXIS is built on the concept of handlers. A handler is an object that performs a specific task. A handler chain is a special type of handler that can contain other handlers and chains.

AXIS framework consists of three chains - transport, global, and service which will process the message in that order, before reaching the service. Figure 1 presents AXIS Engine and their handlers and chains.

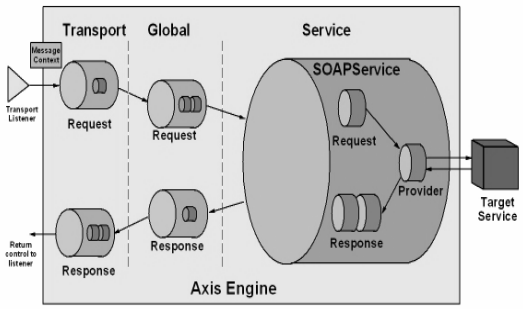

Figure 1. Axis handlers and chains

When the AXIS engine runs a set of Handlers are invoked in a predefined sequence. The particular sequence is determined by two factors: deployment configuration and whether the engine is a client or a server.

When a message is received, a Transport Listener will create a *MessageContext* [10] and will invoke the AXIS processing framework.

Theses handler chains can be configured and new handlers can be added modifying AXIS behavior.

4. SRP Protocol

SRP (Secure Remote Password) is one of the most used password-based authentication protocol. It provides a way to strongly authenticate a user without the usual risks of dictionary attacks faced by other password-based authentication schemes.

In this protocol the password is neither stored as a plain text nor in a ciphered way. Instead, a verifier, obtained from the password through a one-way hash function, is stored.

Another important characteristic of this authentication scheme is that the password is never sent across the network, thus avoiding that an intruder spoofs the network and retrieve the password or some information that could make possible a password reconstruction.

During the authentication process, ephemeral public keys are exchanged between the server and the client and these keys are different for each authentication session.

Another important SRP assumption is that a user can choose a "weak" password without impacting the strength of the authentication scheme.

To perform the authentication a set of handshakes, between the server and the client must be accomplished. Figure 2 shows each step of SRP protocol.

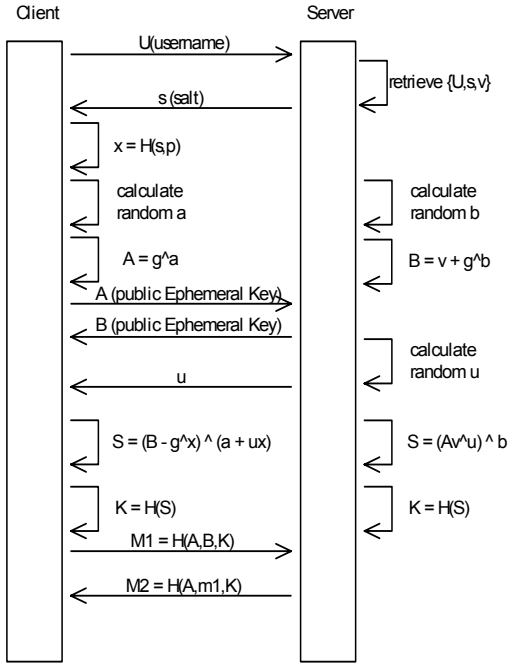

Figure 2. SRP protocol handshake

In first phase [4], the client sends to the server, its username (U). Then the server looks up in its database for the password verifier (v) and the client salt (s). The retrieved salt (s) will be sent across the network to the client. The client salt (s) is randomly generated and the verifier (v) is calculated when the password is stored in the database.

After receiving the client salt it computes a private key (x). This private key is calculated using a one-way hashing function (H) using as input the salt(s) and the password (p). Then the client generates the ephemeral private key (a) and the server calculates its ephemeral private key (b); both keys are generated randomly and not publicly revealed. Using the generator (g) and the ephemeral private key (a), the client calculates its public ephemeral key (A).

In order to exchange its private ephemeral key (B) with the client the server calculates it using the verifier

(v) retrieved from the password database, the generator (g) and its private key (b).

Then the client and the server exchange theirs public ephemeral keys. Using different methods and the public keys exchanged, both, the client and the server, calculates a session key (S). If the password is correct both keys will match.

Both sides hash the session key (S) into a cryptographically strong session key (K).

To show an evidence that the client has the correct session key, it computes the first match value (M1) and send it to the server. Then, the server computes this match value (M1) himself and verifies if it matches with the client's value.

If values match, the server calculates the second match value (M2) as evidence that he also has the correct strong session key.

This second match value (M2) is sent to the client. The client calculates the second match value himself and if both values are equal the client will be successfully authenticated.

Once the user is authenticated, an open secure session could be established for the communications

5. Web Service Authentication Control

Before proposing the solution three basic goals were stated: first, the Secure Remote password protocol should be extended and converted to the Web Service way of work, in order to provide an end-to-end authentication; the authentication issued by the SRP server would be sent as SAML assertions to all Web Service providers required in the service chain and finally the solution should be simple, and easy to be deployed, even with working Web Services that does not provides yet an authentication mechanism.

Considering the goals and the technical characteristics of SOAP, AXIS, SRP and SAML, the solution showed in figure 3 was proposed.

This solution extends the web single sign-on high-level use case provided by the Organization for the Advancement of Structured Information Standards (OASIS) [3] and besides that uses a Web Service perspective to solve the authentication control problem and the use of SAML within Web Services.

In the solution proposed an AXIS handler (*SrpClientHandler*) will be added in the Web Service consumer message flow.

This handler will be part of the global request chain of this consumer, indicating that every time that this consumer requests a service, this handler will be executed before sending its request.

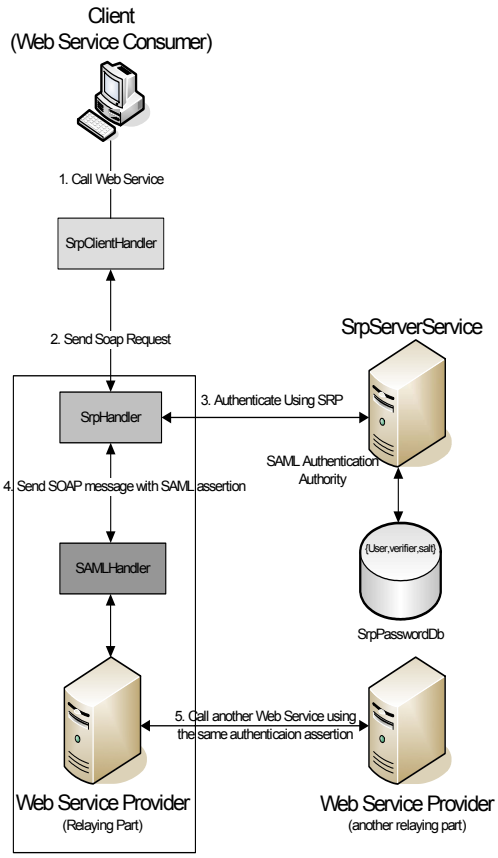

Figure 3. Authentication Handlers

After being called, this handler will add in the original SOAP message a specific header block.

This header block will contain the username and the password that will be used in the authentication process, as shown bellow:

```
<soapenv:Header>
 <srpAuth:SrpAuthentication>
  <SprAuthenticationRequest>
   <user>flavio</user>
   <password>123</password>
  </SprAuthenticationRequest>
 </srpAuth:SrpAuthentication>
</soapenv:Header>
```

Figure 4. SOAP header used in authentication

An important assumption is that that this SOAP message will be transported on a secure channel. This can be easily achieved by using Secure Hypertext Transfer Protocol (HTTPS) in point-to-point communication between the *SrpClientHandler* and *SrpHandler*.

In the same way two other handlers will be added in the global handler chain of the targeted Web Service, *SrpHandler* and *SamlHandler*

By doing this, every time this server receives a request, these two handlers will be invoked, in this order, before the SOAP request processing.

SrpHandler will perform the inverse process as it is the *SrpClientHandler* peer, in other words, it will catch the message and then it will retrieve the header block added by the peer, initially, in the message.

Then, *SamlHandler*, which will be called after *SrpHandler,* will verify if the SOAP request contains the correct SAML authentication statement.

Both in the presence of error during this processing or invalid authentication each handler will throw a <*soap:Fault*> message and the processing will be aborted.

To perform its task *SrpHandler* will receive a *MessageContext* object. This object contains the modified SOAP message.

SrpHandler will verify whether the header block is valid and if it contains the necessary information to authenticate the client.

Getting that the header is valid, *SrpHandler* will then starts the SRP protocol handshaking to verify whether the access will or not be granted.

In the proposed solution, the other peer, which will perform the server side of the SRP protocol, will be another Web Service, so this peer-to-peer communication will be done using SOAP messages, as well.

Figure 5 shows the messages that are exchanged between *SrpHandler* and *SrpServerService*.

The exchanged messages are the basis of SRP protocol and each time the handshake takes place different parameter values are used. This handshake cannot be reproduced elsewhere.

If the information initially received by the *SrpHandler* is correct the last handshake operation – *verify(m2,m2_cli)* – will succeed and the authentication handler will authenticate the user.

In the presence of problem during the authentication process *SrpHandler* will throw a <*soap:Fault*> and the original request will be blocked.

If the authentication is correct the *SrpHandler* will perform a final operation and this operation is not related with the SRP protocol itself.

This operation - *getSamlAuthenticationAssertion()* - is the central point in order to use SAML to exchange with other Web Service providers the fact that this service has been authenticated successful right before.

This method will modify the original SOAP request by adding an authentication assertion to its message body element.

The next handler in the global chain – *SamlHandler* – will now be called; this handler will verify if the SOAP message request contains in its body the SAML authentication assertion. If the assertion is present the handler will not interfere and the SOAP message will continue its original path. In case of absence of the authentication statement this handler will throw a <*soap:Fault*> and the original SOAP request will fail.

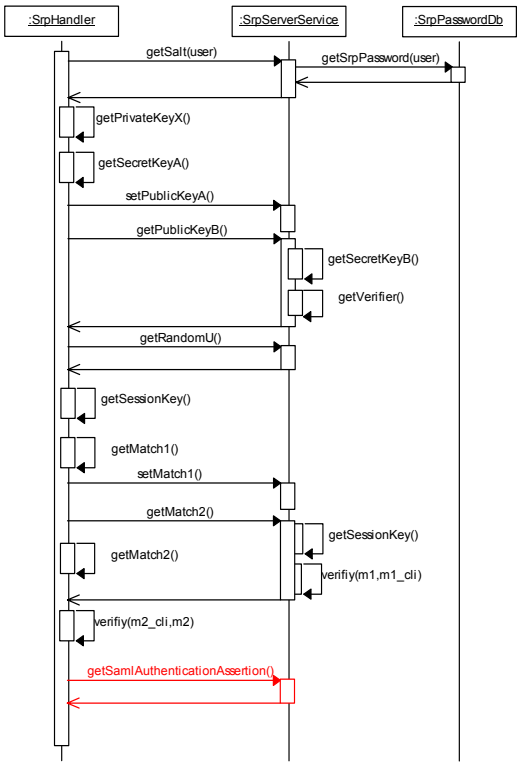

Figure 5. Authentication Web Service calls

After being processed by both handlers successfully the SOAP message will continue its path and will reach the targeted service.

This targeted Web Service can call it its processing other Web Services, which may require an authentication; in this case, if these other Web Services rely on the authentication assertion provided by the *SrpServerService* server, the initial authentication will be exchanged transparently.

The communication between *SrpHandler* object and *SrpServerService* shown in Figure 5 represents, indeed, Web Service calls. *SrpHandler* is located at the targeted Web Service and *SrpServerService* is located at another Web Service host.

To perform these operations the SRP protocol had to be extended and its handshake is now performed between these two Web Services using SOAP messages.

These services and the protocol primitives were defined using the WSDL (Web Service Description Language)[11].

These service definitions describe completely the Web Service and protocol primitives that are Web Service calls.

Although, the proposed idea was based on Apache AXIS architecture, the target Web Service can be any Web Service platform. I.e., this authentication scheme can be added to any Web Service, using an Apache AXIS Web Service as the authentication authority.

6. Handler Tests and Results

The proposed solution was deployed in order to verify if it was based on a correct approach and to test how these handlers could be added to a running Web Service. It was deployed using the latest stable AXIS engine available. To manipulate SAML elements it was used OpenSAML [12]. OpenSAML is a set of open source Java libraries that implements SAML 1.1 specifications.

To work with SRP protocol an object database had to be created. This database (*SrpPasswordDb*) contains objects that specify the authentication information used by SRP (see Figure 2). These objects contain the user name (U), the user salt (s) and a password verifier (v). This information will be added to the database when a new user is added to the system or when its password is modified. A simple application was created just for the database maintenance.

Table 1 shows how a "weak" password as "123" was stored in this database:

Table 1. Password stored information

Username (U)	flavio
Salt (s)	1495651668
Verifier (v)	794475833261338707663717023610959323625378500793796898490895902743448916587325152144362360149735177660234988614249774914394619824624673472788571207151515345373469522257461663172628582595309974034591083070888771145141425854549397348237012950496970919285196756046873246483705961449828584583135796507925500233 89

The first handler, named *SrpClientHandler*, was deployed in the client handler chain. So every time that the client makes a request this handler will be executed. On AXIS it can be accomplished by adding the following lines on a file named "client-config.wsdd", as shown bellow:

```
<globalConfiguration>
  <requestFlow>
    <handler name="srpclient"
      type="java:client.SrpClientHandler">
    </handler>
  </requestFlow>
</globalConfiguration>
```

Figure 6. Client handler configuration

Using Apache AXIS this handler can be easily added using a deployment tool, AdminClient [2], to any application.

After leaving the client, the SOAP message will be sent to the target Web Service, but before reaching this service, a global handler, named *SrpHandler*, will be executed and then another handler, named *SamlHandler* will be executed as well.

These handlers were added to the server handler chain in the same way used in the client and in this case the handler information was added to a file named "*server-config.wsdd*", as shown below:

```
<globalConfiguration>
  <requestFlow>
    <handler name="srp"
      type="java:serviceserver.SrpHandler">
  <parameter name="scope" value="Application"/>
    </handler>
    <handler name="saml"
      type="java:serviceserver.SamlHandler">
  <parameter name="scope" value="Application"/>
    </handler>
</globalConfiguration>
```

Figure 7. Server handler configuration

Considering that the handshake between this handler and the authentication Web Service will be done using multiple Web Service calls, the scope of the handler and the authentication server had to be defined as "Application" indicating that the life time of the objects created during the handshake process will be as long as the Web Service is active.

When a message arrives to the Web Services, this handler (*SrpHandler*) starts the SRP protocol handshake, as shown in figure 5.

In case of a correctly completed authentication, *SrpHandler* will now obtain from the authority assertion (*SrpServerService* server) an authentication assertion.

Initially a SAML request has to be created and sent to *SrpServerService*. This SOAP request will contain a SAML authentication query. Figure bellow shows a simplified view of the message that was sent to *SrpServerService*.

```
<soapenv:Body>
  <samlp:Request MajorVersion="1"
    MinorVersion="1"
    RequestID="200131189660001">
    <saml:AuthenticationQuery>
      <saml:Subject>
        <saml:NameIdentifier
          NameQualifier="facom.ufu.br"
          Format="#emailAddress">
          flavio@facom.ufu.br
```

```
        </saml:NameIdentifier>
      </saml:Subject>
      <saml:AuthenticationMethod
        type="urn:ietf:rfc:2945"/>
    </saml:AuthenticationQuery>
  <samlp:Request>
<soapenv:Body>
```

Figure 8. SAML authentication query request

Considering that service was correctly authenticated the authentication authority responds successfully with the following with the SAML shown in figure 9.

The authentication assertion contains information about authentication that was performed using SRP protocol.

The assertion returned by the *getSamlAuthenticationAssertion()* method will be added to the original SOAP request, other relaying Web Services that will process this authenticated request will use this information.

```
<soapenv:Body>
  <samlp:Response MajorVersion="1" MinorVersion="1
    ResponseID="200131189660002"
    InResponseTo="200131189660001"
    StatusCode="success">
    <saml:Asssertion MajorVersion="1"
      MinorVersion="1"
      AssertionID="200131189660005"
      Issuer="20013118966"
      IssueInstant="2004-06-15T14:06:31Z">
      <saml:AuthenticationStatement
       AuthenticationMethod="urn:ietf:rfc:2945"
       AuthenticationInstant=" 2004-06-15T14:06:31Z">
        <saml:Subject>
          <saml:NameIdentifier
            NameQualifier="facom.ufu.br"
            Format="#emailAddress">
              flavio@facom.ufu.br
          </saml:NameIdentifier>
        </saml:Subject>
      </saml:AuthenticationStatement>
    </saml:Assertion>
  </samlp:Response>
</soapenv:Body>
```

Figure 9. Authentication assertion response

At this time the handler (*SrpHandler*) will them finish its work and the original request sent by the client will continue its path way to the target service.

Finally, before reaching the service, *SamlHandler* will verify if the request contains the SAML assertion, and in case of success, as in this case we are showing the original the Web Service provider will accept request.

6.1 Latency time

In order to verify the overhead produced by the proposed solution its latency time was compared to a system without authentication

Figure 13 shows the measured times. As expected SRP protocol and SAML manipulation introduces an extra processing time. This time was about 4,74 times higher.

In order to measure time required only for the authentication system proposed the network round-trip was not considered, so the same machine hosted the all the components showed in figure 3 – the client, the authentication server and the desired service as well.

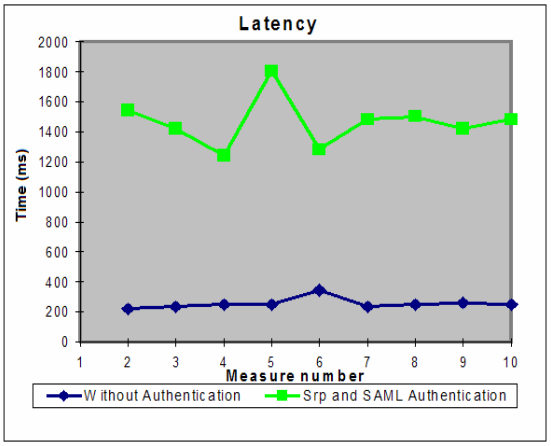

Figure 10. Latency time in milliseconds

7. Concluding Remarks

This paper presents a solution to get an effective authentication control to the Web Services.

SOAP messages were used to extend SRP protocol, and, as far as we know, this is the first implementation of SRP over SOAP protocol.

The extension of SRP protocol presented is easy to be deployed and all the information was treated as a string element ("*xsd:string*"), thus, getting this solution a high level of interoperability between the most distinct Web Services servers.

In this paper we have used SRP-3 [4] protocol to implement the handshakes. The latency time achieved in the round-trip of the service was almost five times greater than the use of the system without the proposed solution. This result was expected concerning the extra processing added to the system. A right improvement to be done is use SRP-6 [13], which contains refinements in SRP protocol. In this version of SRP, the protocol handshake uses less web service call, thus, the latency time tends to be shorter.

An important contribution is the fact that the authentication server is a Web Service as well; in this manner the solution can be easily added and reached in order to provide authentication control to a Web Service and even to legacy applications.

The strong session Key (*K*) created by the SRP protocol can be used as well, to provide a session identifier in order to create a session control of an authenticated user. As a future work, this solution can be extended to provide session management between different information systems using SAML.

The proposed solution, which uses SAML in a Web Service context extends the single sign-on use case [3] proposed by OASIS, this new profile is a valuable contribution because SAML is suitable for the use with Web Services, but OASIS presents a general use case based on HTTP transport only.

Although the tests were made using Apache AXIS the client handler (*SrpClientHandler*) can be constructed to any available Web Service engine, leveraging security and making Web Service authentication control, an essential feature, suitable to any existent deployed Web Service.

Web Services is the interface to interconnect autonomous and heterogeneous systems, distributed geographically, which is the concept of cooperative information systems. In this context security is a critical issue and the need of a strong authentication control is mandatory.

Internet applications focus, initially, was the communication between individuals while Web Services is focusing on providing a communication between applications over Internet in a very straightforward way. To unleash this power to an effective use by cooperative information systems, security plays an import role.

This paper shows how to improve authentication control based on open source technologies, the strong secure remote password (SRP) and SAML that will become shortly a "*de facto*" standard to exchange Web Services access control information.

8. REFERENCES

[1] Booth, D. et al. Web Services Architecture. World Wide Web Consortium Working Draft, 2003. [cited in December 22, 2004]. Available from: <http://www.w3.org/TR/2003/WD-ws-arch-20030808>.

[2] AXIS. Axis User's Guide. [cited in December 22, 2004]. Available from: <http://ws.apache.org/axis/java/user-guide.html>.

[3] Hughes, J. et al. Technical Overview of the OASIS Security Assertion Markup Language (SAML) V1.1 [cited in December 22, 2004]. Available from < http://www.oasis-open.org/committees/download.php/6837/sstc-saml-tech-overview-1.1-cd.pdf>.

[4] Wu. T. The Secure Remote Password Protocol. In Proceedings of the Internet Society Network and Distributed System Symposium, pages 97-111, March 1998.

[5] Skonnard, Aaron. Essential XML Quick Reference: A Programmer's Reference to XML, XPath, XSLT, XML Schema, SOAP, and More, Indianapolis: Pearson Education Inc., 2002.

[6] Englander, R. Java and SOAP, California: O'Reilly & Associates Inc, 2002.

[7] Hartman, B. et al. Mastering Web Services Security, Indianapolis: Wiley Publishing Inc., 2003.

[8] Maler, E. et al. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1 [cited in December 22, 2004]. Available from <http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf>.

[9] Booth, D. et al. Web Services Architecture. World Wide Web Consortium Note. [cited in December 22, 2004]. Available from <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211>.

[10] AXIS. Axis Architecture Guide. [cited in December 22, 2004]. Available from: <http://ws.apache.org/axis/java/architecture-guide.html>.

[11] Christensen, E. et al. Web Services Description Language (WSDL) 1.1. World Wide Web Consortium Note. [cited in December 22, 2004]. Available from <http://www.w3.org/TR/wsdl>.

[12] OpenSAML. OpenSAML 1.0 - an Open Source Security Assertion Markup Language implementation. [cited in December 22, 2004]. Available from: <http://www.opensaml.org>.

[13] Wu, T. SRP-6: Improvements and Refinements to the Secure Remote Password Protocol, Submission to the IEEE P1363 Working Group, Oct 2002. [cited in December 22, 2004]. Available from: <http://srp.stanford.edu/srp6.ps>.