An Arbitration Web Service for E-learning based on XML Security Standards

ROBERT ANDREI BUCHMANN, SERGIU JECAN
Business Information Systems Dpt.
Faculty of Economic Sciences and Business Administration
Babes Bolyai University Cluj Napoca
Str. Th. Mihali 58-60, Cluj Napoca
ROMANIA
robert.buchmann@econ.ubbcluj.ro, sergiu.jecan@econ.ubbcluj.ro

Abstract: This paper promotes a non-repudiation system for student evaluation in an e-learning environment based on web services, AJAX frameworks and PEAR packages in order to implement XML security standards, to provide improved user experience, asynchronous data exchange and message authentication for on-line test papers. The motivation of this research is the need to arbitrate on-line evaluation for both parties, the e-teacher and the e-student, the evaluation criteria and the rating against open answers.

Keywords: - e-learning, XML, digital signature, AJAX

1 Introduction

E-learning already provides a mature model for educational processes. However, the separation of parties in time and space is bound to communication gaps causing situations where an arbitration system should provide trust mechanisms between the student and the teacher.

The proposed application model is an evolutionary step of the XML-driven e-commerce application previously implemented for Flash 2004 with XML support and currently migrated under an AJAX framework and within an e-learning context based on trust managed through XML Signatures. The Flash-XML application model was previously presented in [1] and more details are provided on the mapping of relational data on an XML repository in [2].

2 Problem Formulation

Student evaluation must be treated as a contracting operation: in real life, the test paper is a contract signed by both parties, one providing an evaluation service and signing for the rating and criteria offered with this service, and the other one signing a request for participating to and consuming the evaluation service in a manner similar to atomic transactions - the participation to an exam is fully acknowledged by both parties and cannot be rolled back or replayed against the teaching institution regulations. A test paper is part of a limited number of evaluation processes with respect to the tariffs paid for the inclusion into the e-learning process. The results of a test paper would affect clauses in the studying contract, which could involve limitation of access to more advanced courses and even future payments for subsequent trials to overcome a failed exam.

The practical problem approached by the authors' research effort is the need to arbitrate the evaluation process within an e-learning environment, as established popularly in modern universities. The authors are involved in setting up and conducting distance learning processes based on the e-learning platform provided by the represented institution. Within this context, arose the issue of arbitration in contestation processes, especially in tests based on open questions, but also in multiple choice tests with no immediate answer validation.

The proposed solution is an extension to traditional e-learning application models, regarding the evaluation module, based on XML-based digital signature and improved usability through AJAX frameworks. The solution evolved in two versions: a light version using PHP scripts to build and manage an XML signature repository [3], and a more robust version, with the XML signature repository hidden behind a web service.

3 Problem Solution

3.1 Instrumentation

a. The proposed application model is defined on an **AJAX framework** (Prototype being the chosen framework for the prototype implementation) with a server-side based on PHP scripts that build XML documents signed with XML Signature based on session data and the data provided by the student (questions and answers) and the teacher (proper answers referring questions, the weight of the question in the average grade and the rating resulted from the teacher's evaluation).
AJAX is becoming a mainstream solution for Web applications due to its heavily promoted advantages[4]:

- Increased usability and improved user experience;
- AJAX aligns with the trend of evolving usage patterns in Web applications;
- Parts of the processing effort is moved on the client-side, together with parts of the business tier;
- The development process does not involve commercial software and the application is easier to deploy compared with other Rich Client solutions.

Prototype[1] is one of the most successful framework providing high level JavaScript functions that encapsulate and hide most of the disadvantages of AJAX programming: cross-browser decisions, try-catch structures and heavy syntax. Some of the most relevant advantages brought by the Prototype library are:

- Simplifies the JavaScript syntax for accessing elements by id and accessing their attributes;
- Provides new functions for accessing DOM nodes, based on CSS selectors, in many cases preferable to access by id or tag name;
- Provides new objects for transferring data and encapsulating XMLHttpRequest and cross-browser issues;
- Provides Ruby-inspired iteration functions for operating on arrays;
- Provides a high-level element positioning system, which is the foundation for the Script.aculo.us JavaScript effects library[5];
- Simplifies the access to form and their fields and simplifies frequent operations such as toggling element visibility, field focusing, enabling, disabling, selecting;
- Simplifies the snippet insertions in the user interface, using the Insertion class, an AJAX pattern encouraged by Rails frameworks.

c. The structure of an **XML Signature**, as defined in the W3C recommendation, is the following (cardinality explanation on the right side)[2]:

```
<Signature>
<SignedInfo>
<CanonicalizationMethod>
<SignatureMethod>
(<Reference (URI=)? >            0 or 1 occurrences
(<Transforms>)?                  0 or 1 occurrences
<DigestMethod>
<DigestValue>
</Reference>)+                   1 or more occurrences
</SignedInfo>
<SignatureValue>
(<KeyInfo>) ?                    0 or 1 occurrences
(<Object>) *                     0 or more occurrences
</Signature>
```

The components of the signature are embedded within two elements:

- <SignedInfo> contains the list of resources' locations to be covered by the signature each with preliminary hashes (digest);
- <SignatureValue> contains the definitive hash signature resulted from signing <SignedInfo>.

Additional and optional elements are those providing information on the key (public key or certificate) and various object-resources contained within the signature[6].

Three types of signatures are possible using the XML Signature standard:

- The enveloped signature, contained within the signed document, a type of signature that, in order to be verified, must reference a transformation that removes the signature before hashing its contents;

[1] The library is open source and can be downloaded from http://prototypejs.org

[2] The latest W3C recommendation for the standard is available at: http://www.w3.org/TR/xmldsig-core/

- The enveloping signature, containing the signed document in the Object tag;
- The detached signature, separated from the signed resources, it also guarantees the integrity of online resources against tampering and defacement, since the signature breaks if the signed URIs cannot be dereferenced.

The type of signature used on the "test paper" documents by the proposed application is the **detached** one. This provides a complete separation between the signed and the signature files, whereas enveloping or enveloped signatures are more useful in signing SOAP messages used by the web service version of the application.

An example of mixed signing, with a signature detached to some content, enveloped in other content and enveloping other, based on the algorithm RSA and the hash function SHA1, is the following:

```
<Contract>
<ContractContent Id=”SignedContract”>
.....
</ContractContent>
<Signature Id=”SignedPicture”>
<SignedInfo>
<SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<Reference URI="http://localhost/signedfile.xml">
<DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>.....</DigestValue>                    (1)
</Reference>
<Reference
Type=”http:/www.w3.org/2000/9/xmldsig#Object”
URI="#pic">
<DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>.....</DigestValue>                    (2)
</Reference>
<Reference
Type=”http:/www.w3.org/2000/9/xmldsig#
SignatureProperties” URI=”#Assertions”>
<DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>.....</DigestValue>                    (3)
</Reference>
<Reference URI="#SignedContract”>
<DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>.....</DigestValue>                    (4)
</Reference>
</SignedInfo>
<SignatureValue>.....                               (5)
</SignatureValue>
<Object Id=”pic” MimeType=”image/gif”
Encoding=”http:/www.w3.org/2000/9/xmldsig#
Base64”>
.....                                               (6)
</Object>
<Object>
<SignatureProperties>
<SignatureProperty
Id=”Assertions” Target=”#SignedPicture”>
<ValidUntil>year 2020</ValidUntil>
</SignatureProperty>
</SignatureProperties>
</Object>
</Signature>
</Contract>
```

Element (1) contains the hash value of signedfile. Element (2) contains the hash value for the enveloped resource pic which is a GIF picture converted from binary to Base-64, at (6). Element (3) contains the hash for additional signature assertions, such as the validity time-stamp. Element (4) contains the hash value for the contract that envelops the signature. Element (5) contains the definitive hash (mixed signature) for all the resources. If one Reference does not have an URI attribute, it is supposed to refer the content after the SignatureValue.

<Object> embeds all types of additional information needed to the signature, even the signed resource itself as binary data encoded in a text format (usually Base-64), and declared with a MimeType attribute. Its child, <SignatureProperties>**,** permits the description of the signature functionality, such as a time-frame for signature validity or a carrier identity, hashed as resource number (3). The Target of each property must refer to the signature ID. The recipient must be able to process this information. For compliance with the Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures, the syntax of XML Signature has been extended to XAdEs (XML Advanced Electronic Signature) which provide six additional forms of signatures with additional property elements used to impose and check signer identity,

time stamps, certificate references and other details needed for non-repudiable signatures[3].

Optionally, the <KeyInfo> element separates the trust semantics from the signature structure. The recipient application may use a trust mechanism in order to validate signatures semantically (beyond their schema rules). The trust mechanism must retrieve the key information from the XML parser and check the signature against a trusted set of X509 certificates or public keys.

Compared to traditional PKCS#7 signing, which allows multiple documents to be linked in one document, XML Signature allows multiple signatures on the same document and the same signature for multiple documents. In the proposed application a test paper, structured as an XML document, is partially signed by both student and teacher, each signing the content for which he is responsible. On the other hand, the signature file of a teacher signs all the test papers in which he is involved.

XML Signature also finds multiple uses in the web service version of the application:

- Persistent integrity: in web service environments, integrity protection is present on many OSI levels (SSL for example), but it is not persistent, it only works during communication; persistent integrity is particularly useful in scenarios with multiple signers of the same document;
- Nonrepudiation links signature to identity so the signer (teacher, student) cannot deny the fact that he created the document; the KeyInfo tag of the XML Signature refers the public key of the signer and provides hints to the discovery of the private key; the signature may also involve changing data such as timestamps in order to prevent replay attacks; also, digital certificates may be included and validated using an XKMS (XML Key Management Specifications) trust service (not included at this point in the proposed application);
- Authentication is possible if the KeyInfo data is linked to an identity defined by a digital certificate.

[3] XAdEs Reccomendation: http://www.w3.org/TR/XAdES/

3.2 Application architecture and implementation details

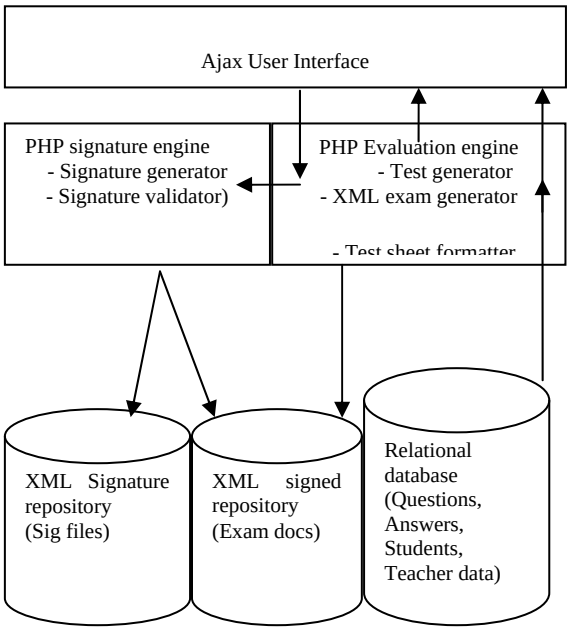

Fig.1. The 3-tier architecture of the application

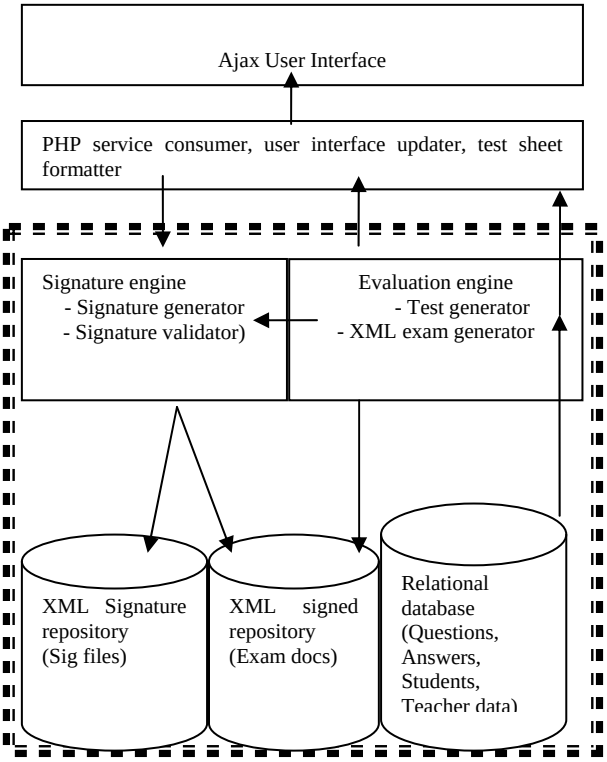

Fig.2. The service oriented architecture of the application

By using an AJAX framework, multiple choice questions can be easily validated and rated in an automated fashion as the respondent is checking his answers, a usability improvement inspired by the

evolved interaction of validated AJAX forms. By handling events such as bluring and focusing, the server provides instant answers to the student, through the Ajax.Updater object provided by the Prototype library:

```
argument={
asynchronous: true,
method:"get",
parameters: data,
onFailure: failurehandler
}
updtr=Ajax.Updater(element,url,argument)
```

where:

- element is the id of the div tag which will have its content replace with the server response;
- url indicates the server script;
- failurehandler is a JavaScript variable of the "function" type, referring a handler; several such handlers may be defined for each state of the XMLHttpRequest transfer and each of them have the request object as an implicit argument.

Ajax.Updater encapsulates the full functionality of XMLHttpRequest, including cross-browser instantiation and page element replacement, combined with the Prototype extended class named Insertion, which provides means of inserting HTML snippets (or formatted XML) at any point of the document relative to other elements or arrays of similar elements (form fields, table rows, div sets etc.). Thus, Ajax.Updater performs both updating of page elements and insertion without defining an event handler (although this is possible, if the arrival of server response has more consequences than the updating/inserting operation).

The instant validation of choice answers is provided by notification mechanisms such as displaying a flashing icon besides the answer, using the Script.aculo.us library of effects:

```
function answercheck(answer, question, notify)
{
data="question="+question+"&answ="+$F(answer)
new Ajax.Updater (notify, "script.php",
        {
        asynchronous: true,
        method: "get",
        parameters: data,
        onCreate: notification,
        })

        function notification(obiectxhr)
        {
        Element.show(notify)
        new Effect.Pulsate(notify)
        }
}
```

However, our main challenge is to define an arbitration system based on integrity preservation and message source authentication during the evaluation process, even in the case of mixed evaluation (open answers and choice answers). By mimicking real-life formalities, both the student and the teacher should be able to sign their input (answers and evaluation data, respectively) and verify each others signature. In the end, the data set built during an exam should have a carbon copy version obtained after both digital signatures were validated on the same data set (document).

XML Signature permits partial digital signatures on element level, in the same XML document or multiple documents. The carbon copy version is obtained in our application model by processing an XSLT stylesheet over the signed "test sheet" document.

The basic use scenario of the application follows the steps:

- The student user fills and submits a test paper presented as an AJAX form, with the possibility of having immediate evaluation on the multiple choice questions;
- The server builds dynamically an XML document with the student's answers and digitally signs it by building an XML Signature via DOMDocument;
- The digitally signed document with the student's answers is stored in an XML repository and can be formatted with an XSLT stylesheet;
- The teacher user is able to verify the authenticity of the student's answers;
- The teacher user signs the part of the document containing his evaluation data (ratings, criteria);
- The student may authenticate the teacher's evaluation;

The digital signature is built with a PHP script by building the XML Signature structures (based on SHA-1 hashing for the prototype in works). Both the student's and the teacher's signatures are stored

in separate files. The student and the teacher's data regarding one "test paper" are stored in the same document, which is partially signed by the two different signatures. XML Signatures allow partial signing of the document (element signing), or multiple signing of the same document.

The following would be the core structure of a test sheet:

examS111T111.xml:

<exam ID="ex1" date="..." student="S111" teacher="T111" course="C111">
<evalset ID="eset1" teachername="John Smith">
<question IDREF="q1"
properA="....." weight="..." rating="..." />
<question IDREF="q2"
properA="....." weight="..." rating="..." />
.....
</evalset>
<questionset ID="qset1" studentname="John Doe">
<question ID="q1" Q="....." A="....." />
<question ID="q2" Q="....." A="....." />
.....
</questionset>
</exam>

The evaluation data set is inserted before the question set in order to increase performance, by minimizing the number of passes for the SAX parsing window (this is kept in perspective, since the current state of the implementation uses DOM). Most elements are using the empty model with attributes also for performance reasons and relational compatibility, as described in the literature and one of our previous studies[7][8].

All exam files are stored by the server in an XML repository with a granularity that defines one document for each exam, and each document uses a pseudo-namespace containing the identifiers of the student and the teacher involved in the exam. These identifiers will be also used to identify their signature files.

Furthermore, each student and teacher has its own file with XML Signature for the list of all their associated elements from the exam documents:

- Teacher T111 has a file with signatures hashing all his evalset elements, in all test sheets;
- Student S111 has a file with signatures hashing all his question set elements.

This is possible since XML Signatures uses double hashing (and consequently, reclaims double validation)[9]:

- Core validation, for the preliminary hashing, applied as a digest of each of the resources in the list of Reference elements;
- Signature validation, for the definitive hashing, applied to the list of preliminary hashes.

The signatures are detached, so they are referencing fragment identifiers from the signed exam, as the following example shows:

student111signatures.xml:

<Signature ID="S111Sign"
xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
<SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
<Reference URI="examS111T222.xml#qset1">
<DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha-1" />
<DigestValue>**.....preliminaryhash.....**</DigestValue>
</Reference>
<Reference URI="examS111T333.xml#qset1">
.....
</Reference>
</SignedInfo>
<SignatureValue>**.....definitivehash.....**</SignatureValue>
</Signature>

teacher222signatures.xml:
<Signature ID="T222Sign">
<SignedInfo>
<CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
<SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#md5"/>
<Reference URI="examS111T222.xml#eset1" >
<DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha-1" />
<DigestValue>**.....preliminaryhash.....**</DigestValue>
</Reference>

<Reference URI=”examS211T222” >
</Reference>
</SignedInfo>
<SignatureValue>**.....definitivehash.....**</SignatureValue>
</Signature>

Both the signature creation and validation are handled by PHP through DOMDocument methods and several encoding functions:

- hash_algos() – returns the list of PHP-registered hashing algorithms (depending on the encryption modules which are installed);
- hash($algo,$string) – applies the chosen algorithm for hashing;
- hash_file($algo,$file) – applies hashing on a file;
- sha1($string) is one of the most used hashing algorithms, preferred against md5() due to superior security;
- base64_encode($string) provides the conversion of hashed binaries to base64 characters that will be actually stored as hash values;
- c14n($string) is a custom function that provides canonicalization of the XML code, in order to obtain a minimal logically unaffected document; canonicalization is important since every bit-level change affects the hashing process and the XML flexibility makes it possible for XML documents that are syntactically different (ex: containing white spaces) to be logically equivalent;
- PEAR packages for signing and verifying raw digital signatures such as Crypt_RSA and Crypt_DSA, the latter being still a proposal package[4]; these provide the value for the SignatureValue element (based on a custom key and formatted through base64 encoding), which is the definitive signature for the canonicalized form of the string obtained from the SignedInfo element.

Based on these functions, the PHP signature engine suggested by Fig.1 provides several high-level functions:

- Signature creation (see code below);
- Adding new test sheets to the signature (opens the signature file and rehashes it);
- Signature verification, by accessing the XML exam file and validating its associated signatures from the teacher and the student.

The next example provides details regarding the signature creation script through DOM ($alg holds the name for the algorithm of choice, currently selected through a form by the signer), for the light version of the application, using PHP scripts instead of the web service:

```
....
if (file_exists('student111signatures.xml'))
$doc->load('student111signatures.xml');
else
{
$doc->formatOutput = true;
$elem=$doc->createElement('Signature');
$root=$doc->appendChild($elem);
$root->setAttribute("xmlns",
"http://www.w3.org/2000/09/xmldsig#");
$root->setAttribute("ID","S111Sign");
$elem=$doc->createElement('SignedInfo');
$si=$root->appendChild($elem);
$elem=$doc-
>createElement('CanonicalizationMethod');
$cm=$si->appendChild($elem);
$cm->setAttribute("Algorithm",
"http://www.w3.org/TR/2001/REC-xml-c14n-
20010315");
$elem=$doc->createElement('SignatureMethod');
$sm=$si->appendChild($elem);
$sm->setAtribute("Algorithm",
"http://www.w3.org/2000/09/xmldsig#".$alg);
$elem=$doc->createElement('SignatureValue');
$sv=$root->appendCild($elem);
$sv->appendChild($doc->CreateTextNode("0"));
}
.........
```

The next steps of the script would be:

- to access the fragment of the exam file to be signed;
- using the functions previously enumerated, to build the hash value for the fragment and the signature for the SignedInfo converted to string; as the standard suggests, the hashes are finally converted to base64;
- to insert through DOM the values as text nodes of the DigestValue and SignatureValue elements.

Signature verification follows the steps:

- opens the exam file and the two associated signature files;

[4] The RSA PEAR package is available at http://pear.php.net/package/Crypt_RSA

- checks the integrity of the signature files, by rehashing its SignedInfo element and comparing the result with the signature;
- checks the integrity of the exam file by rehashing its two parts (identified by ID via fragment identifiers) and comparing them to the preliminary hashes from both parties.

Web services are modular, self-contained applications that can be accessed over a network in order to consume their output on the server-side and, consequently, to render the user interface. Web service traffic raises security issues, since, generally, it is not distinguished by regular HTTP traffic. In the latest development of our application prototype, such a web service would provide, as reusable functionality, the management of evaluation document repository, the signature files, the signed test papers and the database on which the test papers are generated. On the other hand, it implements SOAP-level message security to make the service reusable over any networks (transport security only protects data while in transit). Thus, XML Signature is applied on two levels:

- the document signing;
- the SOAP message signing, through the WS-Security standard;

The service interface permits inputs such as:

- the input of test generation request;
- signature validation request from both parties;

and outputs:

- generates tests;
- validates test signatures for both parties.

According to [10], the web service security must be synchronized with the enforcement of institutional rules governing e-learning activities such as the student evaluation process. Proactive risk avoidance policies are, at least in IT security, much more efficient than reactive measures.

A security architecture must implement the common basics of contract laws and evidence. A law is an agreement giving rise to obligations which are enforced and recognized by that law. A valid contract is comprised by components such as:

- Offer, clear and unambiguous (in student evaluation, this is composed by the test questions, the evaluation criteria and the rating which must be signed and predefined with respect to the evaluation moment);
- Acceptance, final and unequivocal (in student evaluation, this is proved by the signed answers to the test);
- Intention to create a relation between parties, with clearly defined roles (student vs. teacher as actors of a bigger studying contract);
- Consideration or value – the evaluation service has a certain value which was paid for as part of the tuition fee (or even precisely state, for certain type of exams).

Contracting over distributed applications depends on the ability to prove:

- What was agreed: questions, criteria, rating and answers – involves data security, including SOAP message security for the web service;
- When was it agreed: involves inclusion in the signature of a timestamp of each of the involved information in order to prevent replay attacks (resending data);
- Who agreed with it (identity of participants) – involves private key security, with keys stored and allocated according to their roles, to each user account, behind the web service, since key storage is rather an application decision than a standard recommendation.

The contracting itself, in theory, does not depend on any signature; it depends on the wills of the contracting parties. The signature is a mean of proving that the will existed and manifested at some point in time. Additionally, the digital signature also guarantees the contract/message integrity.

The digital signature created by our service involves several operations:

- Preparation of an XML data string (a string of questions generated from a set defined by the teacher, a string of answers related to the questions);
- Preparation of a hashed version of the data string;
- Encryption of the hashed version with a private key of the data creator (teacher, student);
- Both student and teacher can request an identity verification of the data creator established through public key;
- Both student and teacher can request the signature validation by asking the service (through the server script) to rehash their

data and compare it to the hash of the stored test document.

The security implemented over SOAP is based on the WS-Security specification, which defines an XML vocabulary for embedding security tokens within SOAP messages. The WS-Security standard defines the following XML elements:

a. Security, contained in the SOAP header and structured as follows:

```
<S:Envelope>
<S:Header>
<Security S:actor="" S:mustUnderstand=1>
</S:Header>
</S:Envelope>
```

The mustUnderstand attribute states that the recipient of the SOAP message must process the security header, otherwise message processing will fail.

More Security elements may be targeted to multiple receivers, thus the actor attribute establishes the precise relationship between the security token and the service.

b.UsernameToken defines the way of embedding username-password in SOAP, where the data is trasmitted along through a SAML authentication assertion. Example:

c.BinarySecurityToken defines a way of embedding binary data, such as an X.509 certificate. The structure of the element is:

```
<BinarySecurityToken
Id=....
EncodingType=....
ValueType=....>
...encoded value...
<BinarySecurityToken>
```

The attributes provide identity for possible referencing from the SOAP message, the type of encoding for the binary data (Base-64, usually) and the significance of the encoded data (certificate, Kerberos ticket etc).

4 Conclusion

According to [11], the main challenge in e-learning implementations is the actual control that characterizes traditional training. E-learning, as any other e-activity, covers a segment of the Semantic Web pyramid by reaching the higher levels of trust and proof in practical scenarios such as test contestation. The XML Security package of standards provide a flexible set for implementing these levels. Reference is the core element that brings to the XML Signature superior functionality over the traditional binary methods. This element identifies the signed resource and provides the preliminary hash value for signature generation. Its strengths reside in the fact that it can refer any format of data, not only binary and not only XML, either as a detached signature or by referring elements that contain Base-64 encoding of the resource. Thus, XML Signature proves to be a universal signature tool rather than an XML-specific solution. However, it is optimized for **node set** processing, this being the case of any well-formed XML resource. Our research continues on two parallel directions: the development of a full XML Security library for PHP and the potential in e-learning-oriented metadata standards, such as those promoted in [12].

5 Acknowledgment

This paper contains results of the research efforts supported by the following grants financed by the Romanian Research Authority through CNCSIS:

- the doctoral grant TD270/01.10.2007 code 169, managed by Ph. D. Student Jecan Sergiu;
- the research grant 91-049 PNII developed within the author's department.

References:

[1] 1. Buchmann Robert, XML as a Backbone in Web Applications, *WSEAS Transactions on Information Science and Applications*, Issue3, Vol.4, 2007, pp.545-551

[2] 2. Buchmann Robert, XML Databases in E-commerce Applications, *Workshop-ul Informatica Economica si Societatea Informationala*, Timisoara 2006, p.60-65

[3] 3. Buchmann Robert Andrei, Jecan Sergiu, An Arbitration System for Student Evaluation based on XML Signature, *Proceedings of the second European computing conference ECC08*, pp.211-216

[4] 4. Dave Crane, Eric Pascarello, *AJAX in Action*, Manning, 2006

[5] 5. Dave Crane, Bear Bibeault, *Prototype and Scriptaculous in Action*, Manning 2007

[6] 6. Blake Dournaee, *XML Security*, Mcgraw-Hill, 2002

[7] 7. Buchmann Robert, Modeling relational data with XML, *Infobusiness 2006 – Proceedings of the International Conference on Business Information Systems*, 2006, pp.386-395

[8] 8. Buchmann Robert Andrei – *Rolul XML in interoperabilitatea sistemelor informatice pentru afaceri*, Ed. Risoprint, 2007

[9] 9. Buchmann Robert Andrei, Jecan Sergiu, On XML-based digital signature, *Studia Mathematica (Proceedings of the International Conference on Knowledge Engineering, Principles and Techniques)*, 2007, pp.30-38

[10] 10.Mark O'Neill et al. *Web Services Security*, McGrawHill 2003

[11] 11.Chen-Wo Kuo,Quo-Ping Lin, An Empirical Study on Evaluating Government's E-learning to Council for Cultural Affairs, *WSEAS Transactions on Information Science and Applications*, Issue 3, Vol. 4, 2007, pp.472-486

[12] 12. Fu-Ching Wang, Timothy Shih, Pao-Ta Yu, Ming-Tsung Liu, The Development of Metadata Standards for Teaching Domain in Taiwan, *WSEAS Transactions on Information Science and Applications*, Issue 3, Vol. 4, 2007, pp.486-492