Ensuring XML Integrity Using Watermarking Techniques

Romaric Tchokpon, Stelvio Cimato
Dipartimento di Tecnologie dell'Informazione,
Università degli Studi di Milano,
via Bramante 65, 26013, Crema (CR), Italy
{romaric-ange.tchokpon, stelvio.cimato}@unimi.it

Nadia Bennani
CNRS, INSA-Lyon
Université de Lyon,
20 Avenue Albert Einstein, Villeurbanne, France
nadia.bennani@insa-lyon.fr

Abstract—**Today, XML is the most used data interchange format for business-to-business applications. Indeed, an increasing amount of data in XML format is created and published over the Internet every day. Moreover, organizations need more and more to share sets of XML documents usually managed via a common XML repository. XML integrity and authenticity have become strong requirements for applications like web services that exchange messages in such format. XML signature aims to guarantee these properties but it cannot avoid attackers to intercept and change the structure of the XML message. A very common attack to XML Signature called *XML Signature Wrapping(XSW) attack* represents a big issue in web services security as SOAP messages –which are XML signed files- could be corrupted. In this paper, we propose a countermeasure to the XML Signature wrapping attack that makes use of XML watermarking techniques. In our proposal we express constraints on the schema of the XML document and fix its structure using an absolute coordinate system whose values are embedded within the file as a watermark.**

Keywords: XML watermarking; security; SOAP security

I. INTRODUCTION

Today, many applications use the XML format to exchange information. Securing XML data is therefore an important requirement, especially for parameters in transit to and from Web services.

Much more than proof-of-ownership, integrity preservation is a strong requirement in XML-based communications. For XML-SOAP messages exchanged among Web services, it is important to regulate authorized changes to the message structure in order to prevent different types of well-known attacks generally grouped under the term of *XML signature wrapping* [8].

Our proposal focuses on applying XML watermarking to preserve the structure of XML data in transit as a countermeasure to the XML signature wrapping attack. Throughout the paper we will focus on SOAP messages and processing rules, however, the described issues are not necessarily SOAP-specific and can be reapplied in other scenarios.

Digital watermarking is a way to mark robustly a document to prove ownership or copyright properties. Originally, watermarking has been proposed for multimedia documents; later, some researchers investigated the way to apply this technology to other data types such as relational database, text documents and even semi-structured data such as XML documents. There is plenty of works done in digital watermarking for multimedia data (audio, video, images) proving its advantages [14].

Watermarking relational and XML data is quite a recent topic and several challenges remain to be tackled. A major issue is to find a suitable location for the watermark; another one is defining a suitable algorithm for watermark embedding. Also a crucial issue is ensuring the watermark scheme's robustness with respect to modifications of the original document and other types of attacks, as XML files are often updated and reorganized for legitimate reasons. The papers [1,2,4,5] about XML watermarking aim mainly to prove the ownership of XML documents, and embed watermarks by altering some values of the document's content.

In the remainder of this paper we will show how XML watermarking makes it possible to detect and to block any unauthorized structure change. We will use the Schematron language [9] to express constraints on the document schema and embed the absolute positions of each relevant tag of the XML tree within the document.

The paper is organized as follows: In Section 2, we present related works in XML watermarking and the XML signature wrapping attack on SOAP messages. In Section 3, we present in detail our proposal and show how it is an effective countermeasure to XML signature wrapping attack for SOAP messages. Section 4 concludes the paper and discusses future works.

II. Related Work

A. *XML signature wrapping attacks*

The major challenge in XML security is to preserve data integrity. Standard XML Signature techniques contribute to achieve this goal but do not preserve the file against certain attacks.

In 2005, MacIntosh and Austel [8] described a possible attack to XML Signature denoted *XML Signature Wrapping (XSW)*. The aim of this attack is to inject a faked element into the message structure so that a valid signature covers the unmodified element while the faked one is processed by the application logic. The following example presents a sample SOAP signature wrapping attack. To understand how it works, it is important to keep in mind that Web services process incoming SOAP messages in two steps: signature verification first as shown in Figure 1, and service invocation.

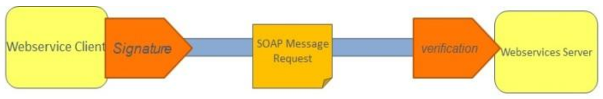


Figure 1.Signature verification for SOAP messages

Figure 2 shows a SOAP message containing a tag address representing a delivery address. The message is valid with respect to the SOAP XML Schema definition. To preserve the field integrity, the tag and its content are signed and the resulting signature is stored in the SOAP message header under the tag *DigestValue* as stated in the security specification [10] while the *@URI* tag contains a reference to the signed address tag in the message.

The attack scenario can be depicted as follows: the attacker intercepts the message before the verification step, moves the delivery address information under the Header by creating a new wrapper and body and she changes the @URL content tag. Finally, she creates a new delivery address under the Body element as represented in Fig. 3.

During the verification process, the server verifies the validity of the message according to the XML SOAP Schema. Then it verifies if all the signed elements and the data they are referred to are present in the message regardless of their position.

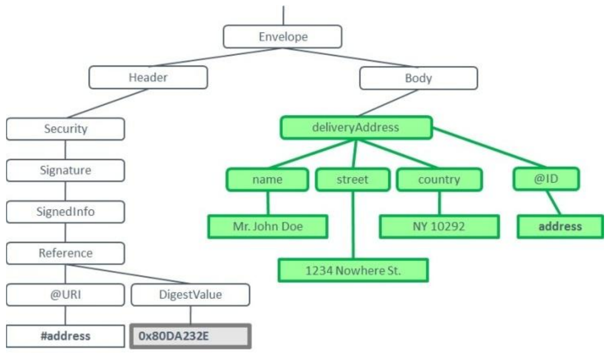


Figure 2. SOAP Request

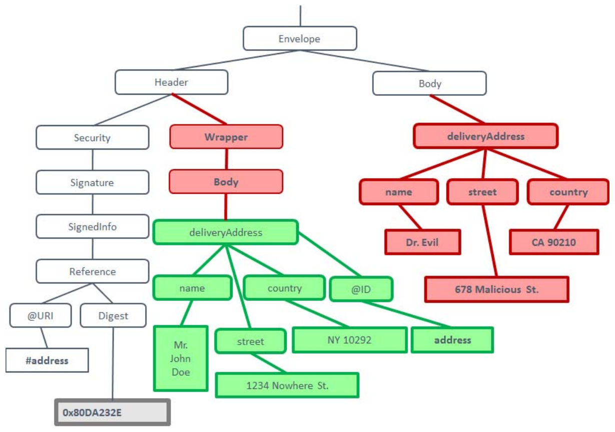


Figure 3. Corrupted SOAP Message

Since the modified message passes all these checks, it will be accepted and the service will be invoked using false data. The Address tag remains well referenced by the @URL tag, the system does not detect any anomaly.

To summarize, SOAP messages signature wrapping attacks are made possible because of the semantic weaknesses of XML Signature about location information. XML Signature protects an element's name, the attributes, and the values without protecting their position in the document. Then, signed XML elements along with the associated signature may be copied to another place while retaining the ability to verify the signature. The order of operations of signature validation is not important from a cryptographic standpoint but it has a significant impact on whether attacks can be performed.

B. *Countermeasures*

Although XML Signature wrapping attack is a quite recent topic, few countermeasures have been proposed and even fewer have been implemented.

The earliest proposals have been exposed in 2005 in [8] where the authors show how to protect against certain wrapping attacks by improving the security policy to be followed by the sender and the receiver. Each message must be validated against an appropriate security policy. This solution, however, does not protect against more sophisticated wrapping attacks. Moreover the proposed security policies are not written in XML syntax, but must be hardcoded into the application. A consequence is a loss in SOA advantages as service can no longer be coupled.

Some countermeasures similar to the previous ones have been proposed in [15, 16], but their proposal failed to protect XML messages simply because the authors didn't cover the signature wrapping attacks semantic.

One proposed countermeasure is to use XML Schema validation [20]. This solution is interesting as validating the XML Schema makes it possible to detect any modification of the SOAP message. However performing schema validation in the Web Services framework is challenging, since it could cause a worsening of the service performance. Furthermore it doesn't guarantee to protect against the signature wrapping attack since XML schemas are extensible.

The third category of proposed countermeasures is called the *inline approach* and was presented in [17]. Instead of forcing the element to be signed by a *WS*-Security Policy as proposed by the previous authors, the authors introduce a way to embed the elements' position in the XML document. The goal of this so-called *in*-line approach is to fix the signed elements so that each movement within the document results in invalid signature verification. This is realized by including a *SoapAccount* element for each referenced element in the XML Signature, holding the number of child elements of the SOAP root element envelope, the number of child elements of the Header element, the number of references in each XML Signature and the successors and predecessors of each signed object.

While the idea of fixing the position seems to be plausible, this solution has some disadvantages: firstly, the *SoapAccount* element is not standardized, thus, the resulting XML Signature is not standards-compliant. Secondly, the saved properties do not prevent XSW in general. An attacker could modify the message structure while fulfilling the secured account information of the inline approach. The idea of embedding the position of important elements of the SOAP message was relevant for our approach. Also operational signature wrapping attacks with the in-line approach countermeasures have been already explored in [18].

Finally many authors agree that the best solution will be the one that will provide a practical mechanism to verify if the structure of the SOAP message is preserved. Our proposal improves the ideas previously proposed and gives a robust mechanism to detect any modification of the SOAP message structure regardless the possible operations.

In our approach, we namely identify a set of constraints that the file structure has to comply to and express them by way of a Schematron [9] assertion and a table of coordinates that fixes the structure of the original document. The coordinates of targeted nodes are stuck within the SOAP message signature using XML watermarking techniques. Any modification can be then detected by retrieving and verifying the nodes coordinates. Details about this technique are given in the next sections.

C. *XML watermarking*

Copyright protection has become a major requirement for many applications and digital watermarking is a widely used technique to achieve this goal. Originally digital watermarking was applied to multimedia documents mainly images, video and audio files. Research about applying digital watermarking to other types of data started some years ago. The first attempt to watermark databases data is proposed in [1] and is related to relational databases. The proposed watermarking algorithm selects the numeric attributes (called *locators*) that are targeted by the watermark, and then chooses the tuples for whom the values of the previously selected attributes will be used to embed the watermark. The watermark embedding impacts the least significant bits.

The proposal is robust against some attacks such as data updates but is applicable only to numerical attributes; however, it does not handle file reorganization.

Zhou and al. in [3] extended the previous solution to XML files. In their solution, locators are XML elements instead of relational database attributes. Considering the tree structure of an XML file, the solution selects only numeric and textual leaf nodes as locators. In [3], a locator selection algorithm has been proposed to identify a set of queries templates to increase data usability. The proposal describes three major features that must be reached by a watermarking schema: *Imperceptibility* meaning that the watermark should not alter data usability; *resilience* meaning that attacks do not succeed to delete the watermark and finally *credibility* meaning that a detection stage will be able to identify the data owner. Besides, in [2] the locator selection procedure is extended to non-leaf nodes. Each selected locator receives one bit of the sequence of bits constituting the watermark to be embedded. In order to be resilient to data reorganization, the verification step attempts to establish a mapping between the original structure of the XML file and the structure of the file to verify. Then, locators are automatically rewritten to be compliant to the new structure in order to retrieve the watermark. The rewriting phase is necessary as it allows localizing the locators in a specified order to recognize the watermark, but it reduces the flexibility and increases the costs of the operation.

Shingo and al. in [4] propose some techniques to embed a binary sequence of bits they call a *pattern* within an XML file. Patterns are used to express the XML data structure. An example is protecting the order in which attributes appear within XML elements. Let us consider an element `elt` with two attributes `att1` and `att2`. The appearance of `att1` then `att2` in the element `elt` corresponds to embedding a '1' while the opposite order (`att2` then `att1`) corresponds to a '0'. This solution is however applicable in a professional use of XML files where strict DTD or XML Schema validity has to be enforced. Besides, this solution looks weak against reorganization attacks.

Finally, Mir and al. in [5] propose a specific watermarking technique for web content. In their approach, a XML based HTML file is watermarked by replacing some words with their synonyms or acronyms, stored in a list. The browser uses the synonym or acronym list to perform the verification stage. This technique is not applicable in other contexts than web browsing; it is however interesting in the sense that it involves the browser in the verification stage.

Summarizing, all previous works focused on static data that are supposed not to change frequently or stored in a database. In the case of XML data in transit like SOAP messages, there is the challenging need to guarantee the integrity of messages. The solution we propose in this paper is aimed to prevent Soap messages signature wrapping attack using XML watermarking techniques.

III. THE PROPOSED SOLUTION

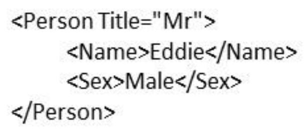
Our solution consists of an embedding and a verification procedure. Embedding consists of three steps. The first one is the expression of constraints on the structure of the XML file (the SOAP envelope in our use case) using Schematron [9] language. The second step is the computing of absolute coordinates of all the nodes in the XML file. In the third step, we identify which coordinates have to be watermarked, in order to successively perform the verification phase. The tags where information is embedded are those, which present a high usability degree [2].

The Verification process consists of localizing the watermark, using approximated XML queries [6][13] then verifying the constraints.

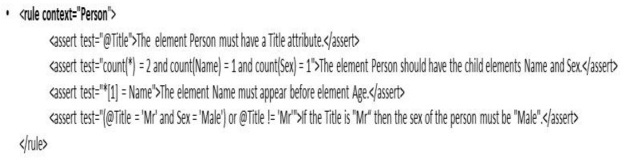
A. Schematron

Schematron, designed by Rick Jellife, [9] is a rule-based language rather different from grammar-based languages like DTD, XML Schema and RELAX NG.

Rule-based approaches take an open method; everything not explicitly disallowed is treated as valid. Schematron is based on finding tree patterns in the parsed document rather than on grammars. Fig. 4 illustrates an example of a Schematron rule applied to the element Person that has one attributes (Title) and two elements (Name and Sex). The rules are expressed in the Rule element that has an attribute Context pointing to the element of the XML file concerned by the rule. Each rule is represented as an assert element and the output of the validation that should appear if the test fails.



```
<Person Title="Mr">
    <Name>Eddie</Name>
    <Sex>Male</Sex>
</Person>
```



```
• <rule context="Person">
      <assert test="@Title">The  element Person must have a Title attribute.</assert>
      <assert test="count(*) = 2 and count(Name) = 1 and count(Sex) = 1">The element Person should have the child elements Name and Sex.</assert>
      <assert test="*[1] = Name">The element Name must appear before element Age.</assert>
      <assert test="(@Title = 'Mr' and Sex = 'Male') or @Title != 'Mr'">If the Title is "Mr" then the sex of the person must be "Male".</assert>
  </rule>
```

Figure 4. Schematron examples

In principle, one could think of using Schematron rules to express the structure of each XML data item to be transmitted, and verify that rules are still satisfied when it is received. But this option is not very effective, as linking between XML data and Schematron expressions is obtained via an external URL added to the XML header. This URL could also be erased or modified by the attacker, as it happens in XML Schema injection attacks. Also, a server with a valid Internet access is needed to enforce the Schematron rules attached to XML data items. This solution is then connection dependent. In our proposal we watermark each XML data item with the absolute position of the elements of the file that are of interest for enforcing Schematron constraints. This can be done by extending existing labeling and indexing techniques for XML files.

B. Absolute position coordinates

We use a coordinate system to get the absolute position of all the elements of the XML file. The tree representation of the XML file is used at this step. In this representation each element is called a node and sub-elements are child nodes. In this case, child nodes of the same parents are *siblings*. Fig. 5 shows the tree representation of a SOAP message.

The second step is to define a coordinate for each node of the tree. Our coordinate system should contain information about the ancestor-descendant or parent-child relationship between nodes. The advantage is that it should be possible to reconstruct the path of each node. There are many techniques to label and index XML trees using a coordinate system [11]. We will use simple Dewey technique [12] to number the nodes.

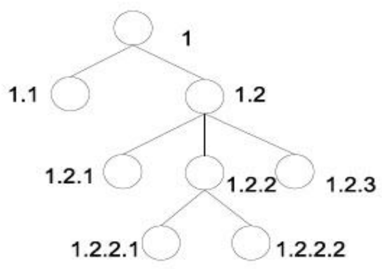


Figure 5. Dewey labeling technique

Dewey's technique makes use of the order among siblings, which we refer to as local order. The coordinate of an element is got by concatenating the label of its parents (parent_label) with its own local order; the obtained coordinates identify uniquely a path from the root to an element.

Given two Dewey labels A: a1, a2…am and B: b1,b2…bn, the following rules can be used to derive structural information from then:

- P1. *Ancestor/Descendant*. A is an ancestor of B if and only if $m<n$ and $a_1 = b_1$, $a_2 = b_2$, …., $a_m = b_m$,
- P2. *Parent/Child*. A is the parent of *B* if and only if and only if *A* is an ancestor of *B* and $m = n – 1$.
- P3. *Sibling*. *A* is the sibling of *B* if and only if $m = n$ and $a_1 = b_1$, $a_2 = b_2$,…, $a_{m-1}=b_{m-1}$, i.e., *A's parent_label* matches *B's parent_label.*

One advantage of Dewey's technique is the facility to reconstruct the path of a node. For example, in Fig. 5, node 1.2 is ancestor of 1.2.2.1 because 1.2 is a prefix of 1.2.2.1. 1.2.2 is the parent of 1.2.2.1 because 1.2.2 matches the *parent_label*. A second advantage is that through this coordinate system, it is possible to detect elements pertaining to the same level in the arborescence.
A main drawback of this technique is its rigidity. The structure of the tree is fixed and there is no possibility to make any change. It could not be suitable in the case of data frequently updated but it is well suited to our proposal whose purpose is preventing unwanted modifications. Dynamic indexing techniques that could handle legitimate modifications introduced by authorized intermediaries while still preventing wrapping attacks will be investigated in future works.

In the case of our sample SOAP message, the computation of the coordinates according to the Dewey technique is shown in Table 1.

TABLE 1.COORDINATES TABLE OF ORIGINAL SOAP MESSAGE

Nodes	**Coord**	**Nodes**	**Coord.**
Envelope	1	SignedInfo	1.1.1.1.1
Header	1.1	Mr.John Doe	1.2.1.1.1
Body	1.2	1234Nowhere St	1.2.1.2.1
Security	1.1.1	NY10292	1.2.1.3.1
DeliveryAddress	1.2.1	**address**	**1.2.1.4.1**
Signature	1.1.1.1	Reference	1.1.1.1.1.1
Name	1.2.1.1	@URI	1.1.1.1.1.1.1
Street	1.2.1.2	**DigestValue**	**1.1.1.1.1.1.2**
Country	1.2.1.3	#address	1.1.1.1.1.1.1.1
@ID	1.2.1.4	0x80DA232E	1.1.1.1.1.1.1.2.1

By computing the coordinates of the elements in the corrupted message we get the Table 2.

TABLE 2. COORDINATES TABLE OF CORRUPTED SOAP MESSAGE

Nodes	**Coord**	**Nodes**	**Coord.**
Envelope	1	CA 90210	1.2.1.3.1
Header	1.1	Reference	1.1.1.1.1.1
Body	1.2	Name	1.1.2.1.1.1
Security	1.1.1	Country	1.1.2.1.1.3
Wrapper	1.1.2	@ID	1.1.2.1.1.4
Deliv-eryAddress	1.2.1	@URI	1.1.1.1.1.1.1
Signature	1.1.1.1	**Digest**	**1.1.1.1.1.1.2**
Body	1.1.2.1	Mr John Doe	1.1.2.1.1.1.1
Name	1.2.1.1	Street	1.1.2.1.1.2
Street	1.2.1.2	NY 10292	1.1.2.1.1.3.1
Country	1.2.1.3	**address**	**1.1.2.1.1.4.1**
SignedInfo	1.1.1.1.1	#address	1.1.1.1.1.1.1.1
Deliv-eryAdress	1.1.2.1.1	0x80DA232E	1.1.1.1.1.1.2.1
Dr. Evil	1.2.1.1.1	1234 Nowhere St.	1.1.2.1.1.2.1
678 Malicious St.	1.2.1.2.1		

We can see that the coordinate of the node *address* in Table 2 has changed, which proves the constraint forgery.

C. *The watermarking method*

One issue in watermarking XML file is the choice of the place where to insert the watermark, called the locators. Many criteria could be used to choose the locators. One criterion to choose them is the usability of the corresponding data [2] to obtain file coherency. This requirement means that the removal of this tag from the file will break its meaning. Another possibility is also to define a metric to evaluate the frequency of each tag in the document and then choose the less or most frequent tag; the less frequent because the probability that a malicious user modify this tag is low, as it has not enough importance in the file. We shall choose an embedding location, ID-IDREF pair of attributes, which are burdensome to change without affecting data usability.
The next step is to choose the watermark embedding method. Instead of adopting techniques like in [2] where the targeted content is altered due to watermark embedding, we propose to embed the watermark in a locator whose value is independent of the data in the XML file. A good example is to embed the watermark in the ID and IDREF attributes of well-chosen tags in the XML file.

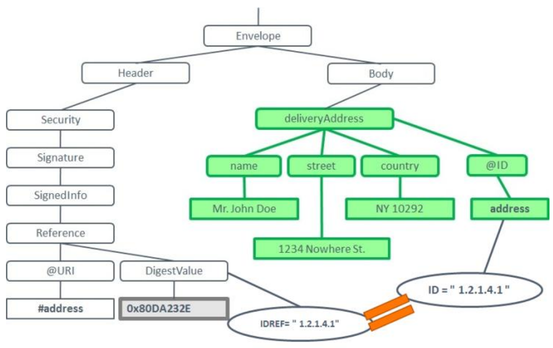


Figure 6. SOAP message with ID and IDREF

D. *The watermark value*

The value to embed is the coordinate of the XML node that has to remain unchanged in the XML tree structure. In the case of the SOAP message [10], we will choose the *DigestValue* tag as a target tag and add to it an ID attribute to insert the watermark.

As an additional precaution, as ID tags could undergo modification or deletion attacks, the watermark could also be integrated in the signature. Let's assume *hash* is the hash function, *c*, the coordinate of the *address* element to be signed, and *v* the address value, *DigestValue = hash (c+v),* where "+" is the symbol of concatenation.

For example in the case of SOAP message, *DigestValue = hash(address+***1.2.1.4.1).** The result of this computation is the new value of the element *DigestValue*.

Adding this new value to the signature secures our solution against attributes values changing; however, changing ID-IDREF pairs would come at a cost for the attacker, as it must be done in such a way to preserve the well-formedness (and therefore, parsability) of XML data In all cases, any modification to the value of the coordinates will generate a non-matching *DigestValue* and we can suspect piracy.

E. *The watermark reconstruction*

The first operation of the watermark extraction process is to validate the watermarked XML data item against its XML Schema. Secondly we search into the file for the elements with ID value equal to the IDREF value. A very easy and powerful method is to use fuzzy queries. In [10] we explain how fuzzy queries could be used to extract all the locators from XML file regardless of its structure. In the case of the SOAP message for example, the fuzzy query to find the element address is:

/Envelope{/#}[{@id NEAR '**1.2.1.4.1**'}]

Then, the position of the elements in reference is checked using the coordinate system. If the position is the good one, it is only necessary to compute the digest of the concatenation of this position and the value of the referenced element to check the equality between the existing digest value in the file and the obtained one. The reason of this comparison is that an attacker can change the position of the referenced element and writing new values into the ID and IDREF. If only the position is checked, then the file will be accepted with the new position. By computing the new digest, the two digest values will be different and we could detect the modification of the positions.

IV. Conclusion

In this paper we have presented a new approach for tackling unauthorized modifications of XML data in transit, including the XML Signature wrapping attack. Our solution is based on watermarking XML data items, specifying the absolute coordinates of the nodes whose position in the file structure has to remain unchanged. The coordinates are obtained using the Dewey numbering technique. The obtained coordinates are embedded in the most useful tags using watermarking techniques that do not alter the content of the initial XML file. The expressed structure constraints are firstly expressed using Schematron assertions and then transcribed in a set of coordinates embedded in the XML file. The advantage of embedding such constraints allows to stick them in a lightweight manner to the file and to enforce them even in case the Schematron is impossible to access due to network or server failure.

As a future work we plan to enlarge the set of constraints that could be taken into account using our solution. The other objective is to apply such a method and in the meantime let the XML file structure as flexible as possible using a less rigid ordering and labeling method to obtain the coordinates.

References

[1] R. Agrawal and J. Kierman. "Watermarking relational databases". In Proceedings of the 28th International Conference on Very large Databases (VLDB), 2002.

[2] Xuan Zhou, HweeHwa Pang and Kian-Lee Tan. "Querying query-based watermarking for XML data". In Proceeding ASIACCS '07 the 2nd ACM symposium on Information, computer and communications security ACM New York, NY, USA ©2007.

[3] Xuan Zhou, HweeHwa Pang, Kian-Lee Tan, Dhruv Mangla. "WmXML: A System for Watermarking XML Data". In Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005.Y.

[4] Shingo, Kyoko, Ichiro, Osamu. “A Proposal on Information Hiding Methods using XML”. Mitsubishi Research Institute, Communication Research Laboratory, Yokohama National University and the University of Tokyo, unpublished

[5] Nighat Mir, Sayed Afaq Hussain. “Web Page Watermarking: XML files using Synonyms and Acronyms”. In Proceedings of Conference WASET 2011, World Academy of Science, Engeniering and Technology, january 25-27, 2011.

[6] Ernesto Damiani, Stefania Marrara, Gabriella Pasi. “Fuzzyxpath: using an ir features to approximately query XML documents”. in Proceedings IFSA'07, pages 199-208, 2007.

[7] Niels Agne Nordbotten, "XML and Web Services Security Standards”, IEEE Communications survey and Tutorials, Vol. 11, no.3, 2009, pp. 4-18.

[8] Michael McIntosh, Paula Austel, “XML Signature Element Wrapping Attacks and Countermeasures”, in SWS '05: Proceedings of the 2005 workshop on Secure Web Services

[9] Schematron, www.schematron.com

[10] SOAP Security Extensions: Digital Signature, http://www.w3.org/TR/SOAP-dsig/

[11] B. Catania, A. Maddalena and A. Vakali, “XML Document Indexes: A Classification”, IEEE computer society, September/October 2005, pp. 64-71

[12] Li Ying; Ma Jun; Sun Yuyin; , "Applying Dewey Encoding to Construct XML Index for Path and Keyword Query," Database Technology and Applications, 2009 First International Workshop on , vol., no., pp.553-556, 25-26 April 2009

[13] Romaric Tchokpon, Nadia Bennani, Ernesto Damiani.”Robust XML Watermarking Using Fuzzy Queries”, in SAPSE 2012 in conjunction with COMPSAC, Izmir, Turkey 2012 (to be published)

[14] Fabien A. P. Petitcolas, Ross J. Anderson and Markus G. Kuhn. “Information hiding–a survey”. Proceedings of the I.E.E.E., 87(7):1062–1078, July 1999.

[15] Bhargavan K., Fournet C., Gordon A.D., “Verifying policy-based security for Web Services. In CCS ’04: Proceedings of the 11th ACM Conference on Computer and communication security (New York, NY, USA 2004), ACM Press, pp.268-277.

[16] Bhargavan K., Fournet C., Gordon A.D, O’Shea G. “An Advisor for web services security policies. In SWS’05: Proceedings of the 2005 workshop on Secure web services (New York, NY, USA, 2005), ACM Press, pp. 1-9.

[17] Rahaman M. A., Schaad A. “Soap-based secure conversation and collaboration. In ICWS (2007), pp.471 - 480.

[18] Gajek S., Jensen M., Liao L., Schwenk J. “Breaking and fixing the inline approach. In SWS (2007), pp. 37 – 43

[19] Gajek S., Jensen M., Liao L., Schwenk J. “Analysis of Signature wrapping attacks and countermeasures. In ICWS (2009), IEEE, pp. 575-582.

[20] M. Jensen, C. Meyer, J. Somorovsky, and J. Schwenk. “On the effectiveness of xml schema validation for countering xml signature wrapping attacks”. In Securing Services on the Cloud (IWSSC), 2011 1st International Workshop on, pages 7 –13, September 2011.