IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661, ISBN: 2278-8727Volume 7, Issue 1 (Nov. - Dec. 2012), PP 40-46
www.iosrjournals.org

Security Improvement of Xml Files Using Structure Modification

Daxesh Panchal[1], Prof A.S. Dhabariya[2]
Department of Computer Science, Shrinathji Institute of Technology and Engineering, India

Abstract :*XML is an extensible markup language, with its influential explanatory, scalable, ordered, platform-independent features are widely used in the presence of a large number of e-commerce and e-business data exchange.XML is also broadly used in current distributed systems. The security of the XML based communication, and the Web services themselves, is of great implication to the overall security of these systems. Moreover, in order to assist interoperability, the security mechanisms should preferably be based on established standards. Encryption provides confidentiality by protecting sensitive information from being read by intruders. In this paper, we present one approach for encrypting xml file using separation of xml data and content of xml element.*

Keywords *- Security, XML, Web services, SOA, distributed systems*

I. INTRODUCTION

Security has always been crucially significant in the any company to guarantee the integrity of data and operations, to preserve confidentiality, and to make sure information is used properly. Nevertheless, in today's web-based commercial application, the way for providing that security has changed. In former days we were used to keep all computer resources in a room and lock the room but this kind of security is not enough in present business organizations. Rather than providing physical security, some software, encryption methodology is requiring which protect data and operations of business applications. Economic globalization leads to complex decentralized company structures calling for the broad use of distributed IT-systems. In particular, one has to reflect a new class of security aspects which are relevant to e-commerce-based But not to traditional business processes. Security of computer-based systems mostly concerns confidentiality, integrity, and availability aspects. Confidentiality targets to keeps data secret from unauthorized users. Integrity prevents unauthorized changes in data and operations .Availability ensures that information is available when needed.

II. OVERVIEW OF XML SECURITY

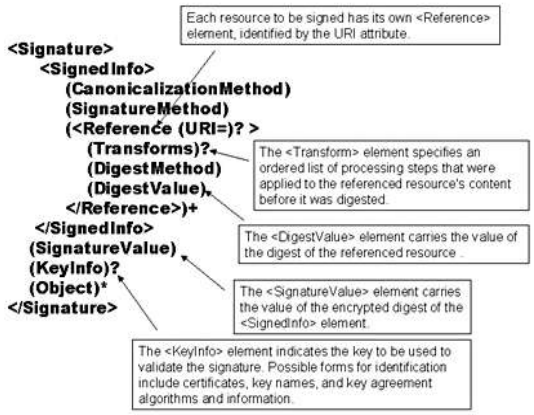
An essential prerequisite of new Internet-wide security standards is that they apply to content created using extensible markup language (XML).XML has been adopted widely for a enormous diversity of applications and types of content. XML is also at the basis of interoperability protocols used to integrate applications across the Internet, such as Web services protocols: the Web service technology relies on different XML-based languages such as Simple Object Access Protocol (SOAP), Web Service Definition Language (WSDL), and Universal Description Discovery and Integration (UDDI) .Xml Security is divided in to two parts.XML Signature and XML Encryption

.

2.1 XML SIGNATURE:

The use of digital signatures is a common method for ensuring message integrity, authentication, and non-repudiation [1].XML Signature defines a standard interoperable format for representing digital signatures in XML and provides mechanisms for efficiently applying digital signatures to XML resources. XML Signature is not limited to signing XML resources, however, as it can also be used to sign binary resources such as a JPEG-file. A single XML signature may cover several resources, where each resource may be an XML document, a part of an XML document or a binary resource.

Signature validation requires that the data object that was signed be accessible. The XML signature itself will generally indicate the location of the original signed object. This reference can be referenced by a URI within the XML signature; reside within the same resource as the XML signature (the signature is a sibling);



The Components of an XML Signature (Fig-1)

Be embedded within the XML signature (the signature is the parent); Have its XML signature embedded within itself (the signature is the child).

2.2 XML ENCRYPTION:

XML Encryption [2] provides confidentiality by allowing selected parts of, or an entire, XML document to be encrypted. XML Encryption is similar to XML Signature in many ways. For instance, like XML Signature, XML Encryption does not apply only to XML resources as it may be used to encrypt arbitrary binary resources as well. Data that is encrypted using XML Encryption is represented by an EncryptedData element. As can be seen, the CipherData element is the only mandatory child element of EncryptedData. CipherData either contains or provides a reference to the cipher text of the encrypted data. As may be noticed, this is equivalent with the enveloping and detached variations of XML Signature. Contrary to XML Signature, however, a single EncryptedData element can only contain or reference one resource. If multiple resources are to be encrypted within the same XML document, multiple EncryptedData elements must be used.

2.2.1ENCRYPTING AN XML ELEMENT

Jay's credit card number is secure information! If the application would like to keep it secret, It can encrypt that element [5].

```
<? xml version='1.0'?>
<MoneyData xmlns='http://MoneyEx.com/MoneyD2'>
<Firstname>Jay</Firstname>
<EncryptedData  Type='http://www.w3.org/2001/04/xmlenc#Element'
xmlns='http://www.w3.org/2001/04/xmlenc#'>
<CipherData>
    <CipherValue>A23B45C56</CipherValue>
</CipherData>
   </EncryptedData>
</MoneyData >
```

By encrypting the CreditCard element from its start to end tags, we are able to hide its identity. Now attackers will not getting any information from this encrypted element.

2.2.2ENCRYPTING XML ELEMENT CONTENT (ELEMENTS)

Another approach is to hide data or children elements, sometimes we wants to allow some information to visible like card expiry date and hide information like its card number [5].

```
<? xml version='1.0'?>
<MoneyData xmlns='http://MoneyEx.com/MoneyD2'>
<Firstname>Jay</Firstname>
<CCard Expiry='5/17' format='MM/YY'>
<CNo>
<EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
xmlns='http://www.w3.org/2001/04/xmlenc#'>
      <CipherData>
      <CipherValue>A23B45C56</CipherValue>
      </CipherData>
      </EncryptedData>
```

```
</CNo>
</CCard>
</MoneyData >
```

Both CCard and CNo are in the clear, but the character data content of these elements are encrypted.

2.2.3 ENCRYPTING ARBITRARY DATA AND XML DOCUMENTS

If the application needs to be encrypt the whole document. This applies to arbitrary data including XML documents [5].

```
 <? xml version='1.0'?>
<EncryptedData Type="http://www.w3.org/2001/04/xmlenc#' MimeType='text/xml'>
   <CipherData>
     <CipherValue>A23B45C56</CipherValue>
   </CipherData>
 </EncryptedData>
```

III. ALGORITHMS AND STRUCTURES [5]

There are various algorithm is available for encryption and decryption as discussed here.

3.1 BLOCK ENCRYPTION ALGORITHMS

Block encryption algorithms are designed for encrypting and decrypting data in fixed size, multiple octet blocks. Their identifiers appear as the value of the Algorithm attributes of Encryption Method elements that are children of EncryptedData. Block encryption algorithms take, as implicit arguments, the data to be encrypted or decrypted, the keying material, and their direction of operation. For all of these algorithms specified below, an initialization vector (IV) is required that is encoded with the cipher text.

3.1.1 PADDING

Since the data being encrypted is an arbitrary number of octets, it may not be a multiple of the block size. This is solved by padding the plain text up to the block size before encryption and unpadding after decryption. The padding algorithm is to calculate the smallest non-zero number of octets, say N that must be suffixed to the plain text to bring it up to a multiple of the block size. We will assume the block size is B octets so N is in the range of 1 to B. Pad by suffixing the plain text with N-1 arbitrary pad bytes and a final byte whose value is N. On decryption, just take the last byte and, after sanity checking it, strip that many bytes from the end of the decrypted cipher text.

For example, assume an 8 byte block size and plain text of 0x616263. The padded plain text would then be 0x616263????????05 where the "??" bytes can be any value.

3.2 STREAM ENCRYPTION ALGORITHMS

Simple stream encryption algorithms generate based on the key, a stream of bytes which are XORed with the plain text data bytes to produce the cipher text on encryption and with the cipher text bytes to produce plain text on decryption. They are normally used for the encryption of data and are specified by the value of the Algorithm attribute of the Encryption Method child of an EncryptedData element.

IV. PROPOSED WORK: STRUCTURE-BASED IMPLEMENTATION

These are the various methods available for encrypting xml document .We would like to use another approach for xml encryption. This approach is for encrypting entire xml document .we would like to separate tag and content of xml element then separately encrypting both data with different symmetric key and then merge encrypted data and generate new xml file.

Proposed Algorithm

For Encrypting Xml file these steps should be performed.

1. Separate xml tag and content from xml file.
2. Encrypt xml tag and content of xml element with different symmetric key.
3. Encrypt these two symmetric key and Store these two symmetric key in two different file with different name on server.
4. Combine these encrypted tag and content of xml file in to one xml file with single <EncryptedData> Element.

For Decrypting Xml file these steps should be performed.

1. Load Symmetric keys from file which reside on server and decrypt it.
2. Separate encrypted tag and content of xml file.
3. Use appropriate keys for decrypting tag and content of xml file.
4. Generate original xml file using tag and data.

Original xml File:

```
<?xml version="1.0" encoding="UTF-8" ?>
<PurchaseOrder>
<Item number="130046593231">
  <Description>VideoGame</Description>
  <Price>10.29</Price>
  </Item>
 <Buyer id="8492340">
  <Name>MyName</Name>
  <Address>
  <Street>OneNetworkDrive</Street>
  <Town>Burlington</Town>
  <State>MA</State>
  <Country>UnitedStates</Country>
  <PostalCode>01803</PostalCode>
  </Address>
  </Buyer>
 <Buyer id="8492340">
  <Name>MyName</Name>
  <Address>
  <Street>OneNetworkDrive</Street>
  <Town>Burlington</Town>
  <State>MA</State>
  <Country>UnitedStates</Country>
  <PostalCode>01803</PostalCode>
  </Address>
  </Buyer>
</PurchaseOrder>
```

This is original xml file now we are going to separate xml tag and content of xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<PurchaseOrder>
<Item number="130046593231">
<Description>#</Description>
<Price>#</Price>#</Item>
<Buyer id="8492340">
<Name>#</Name>
<Address>
<Street>#</Street>
<Town>#</Town>
<State>#</State>
<Country>#</Country>
<PostalCode>#</PostalCode>
#</Address>#</Buyer>
<Buyer id="8492340">
<Name>#</Name>
<Address>
<Street>#</Street>
<Town>#</Town>
<State>#</State>
<Country>#</Country>
<PostalCode>#</PostalCode>
#</Address>#</Buyer>
<Buyer id="8492340">
<Name>#</Name>
<Address>
<Street>#</Street>
<Town>#</Town>
<State>#</State>
<Country>#</Country>
```

<PostalCode>#</PostalCode>
#</Address>#</Buyer>
#</PurchaseOrder>

Tag of XML file.

In above file we have used '#' symbol for representing place for inserting content. Means we can use any special character while separating element, this special character is useful for regenerating xml file.

Let us see the content of xml file.

VideoGame
10.29
James
OneNetworkDrive
Burlington
MA
UnitedStates
01803
Joy
Alkapuri
Vadodara
Gujarat
India
301803
Rajan
AjwaRoad
Vidhyanagar
Gujarat
India
303803

Now encrypt this tag and content of xml file.

This is an encrypted content of all tag.

Tag:-

nD9Llba3WnYO/StcLIxn7d3ygqy78Rw1V1mknZzSJd0W1I+HGw5mfbMndU62Muz8BJ1aBlWzkloU8zF3hm
kc3js4OqWjAvCFMZzgt5cWXZTmepsORRb2LUjlT2JziOhELdg9PVdeYWJ4fsEPVeKBKHEE7xmNDMTJL
V9wGn6NHpCw9uSwcXoZ/sPmnp92C6+ao3vUCh0i0SqrFmtfaePWG69nn6q2XnFVE9T6cGLxQB0jXmnhv+
/MhRRgRVmMH9OZUf9W02Vwt4daoncUp1hbLcvsC647fOoYCXzQm6EYDMHfaPfQKnK1GEUfgNOFUF
Zjxy+FAYwo6qFmkw0H+782JNVxszdeIp3k4jKfMfgv6803LdBkp4ZvBotMx5Te
QOSil+hOBGOEpxZx79RgHdD/YSKhici/JyO9OoTxsGdhtA4gKQGmtr1TjFz5EGg5+hnO11CsMXQyhp2k0L
b1V605xkO1Ii7+QQQGuD7DjGYmIPwIxWcW/O6dJYhQr3XT99558ECGcEDvOZDNPELsJrJL/0+yyMENf8
KOp7jE9pGvTp1tj3nMyiQ/3WNT2HGTPDnczDzNxOjG6g7q/SZNry3Nkpr0JMZlxx+1NTsy4k3yyjIytNAVV
Hsqn8lX1MGDYgXwvCCSYkmeitRA4PTzwnYi9JTzPWNeBZDf2WGGdJ2uWhIDE+GtrpWwkjIRjdAKTp4/
RpzPoyN/GhomMSTuzvGIwwu/bYfBl/lYn1BsFljrzxDIIsravjMoWs0Ky+18zM0qZpMNB/u/NiQZ6SeOij8Aok
dDpQiob/hbp70qjlEjgfezJ3VOtjLs/GLblOU4C1bj.

This is a encrypted content of all tag.

Content:-

5XlBW/roNaMQg6bcpym+ufUiry6gIHy08d/NqvecdKTXY1fn5XPQb60ISOIf4UZd2Z9aYCnATyWxNglBtJF
0TWYHCzRSUo0hJy1wFHXNxJnEa7PzVJzm+cbwFmZktFrmlb9h3pKFhRI+xn1+I6XU+UDFMjXxw3EUG
DxCJfDuMyZ9mr8AdRKq2JHzNkHEXh8+dQef6DtBfp2980PLI0m50NTAlSZY5+z95RxZzU8dev8dWhIfZal
LtuIPhTIbQ/Zv6V5XO29hEWATjXEsJ+OTX7vyfe1eZMmOu/J97V5kyY678n3tXmTJjrvyfe1eZMmOu/J97V
5kyY678n3tXmTJjrvyfe1eZMmOu/J97V5kyY678n3tXmTJjrvyfe1eZMmOu/J97V5kyY678n3tXmTJjrvyfe1eZ
MmOu/J97V5kyY678n3tXmTJjrvyfe1eZMmOu/J97V5kyY678n3tXmTJjrvyfe1eZMmOu/J97V5kyY678n3tX
mTJjrvyfe1eZMmOu/J97V5kyY678n3tXmTJjrvyfe1eZMmOu/J97V5kyY678n3tXmTJjrvyfe1eZMmOu/J97V5
kyY678n3tXmTJjrvyfe1eZMmOu/J97V5kyY678n3tXmTJjrvyfe1eZMmOu/J97V5kyY678n3tXmTJjrvyfe1eZ
MmOu/J97V5kyY678n3tXmTJjrvyfe1eZMmOu/J97V5kyY678n3tXmTJjrvyfe1eZMmOu/J97V5kyY678n3tX
mTJjrvyfe1eZMmOgOH0r/1yaos=

This is a content of all xml elements. Now we would like to combine this tag and content of xml file and generate xml file.

<EncryptedData>nD9Llba3WnYO/StcLIxn7d3ygqy78Rw1V1mknZzSJd0W1I+HGw5mfbMndU62Muz8BJ1a
BlWzkloU8zF3hmkc3js4OqWjAvCFMZzgt5cWXZTmepsORRb2LUjlT2JziOhELdg9PVdeYWJ4fsEPVeKBK
HEExmNDMTJLV9wGn6NHpCw9uSwcXoZ/sPmnp92C6+ao3vUCh0i0SqrFmtfaePWG69nn6q2XnFVE9T6c
GLxQB0jXmnhv+/MhRRgRVmMH9OZUf9W02Vwt4daoncUp1hbLcvsC647fOoYCXzQm6EYDMHfaPfQnK
1GEUfgNOFUFZjxy+FAYwo6qFmkw0H+782JNVxszdeIp3k4jKfMfgv6803LdBkp4ZvBotMx5TeQOSil+hOB
GOEpxZx79RgHdD/YSKhici/JyO9OoTxsGdhtA4gKQGmtr1TjFz5EGg5+hnO11CsMXQyhp2k0Lb1V605xkO
1Ii7+QQQGuD7DjGYmIPwIxWcW/O6dJYhQr3XT99558ECGcEDvOZDNPELsJrJL0+yyMENf8KOp7jE9pG
vTp1tj3nMyiQ/3WNT2HGTPDnczDzNxOjG6g7q/SZNry3Nkpr0JMZlxx+1NTsy4k3yyjIytNAVVHsqn8lX1M
GDYgXwvCCSYkmeitRA4PTzwnYi9JTzPWNeBZDf2WGGdJ2uWhIDE+GtrpWwkjIRjdAKTp4/RpzPoyN/G
homMSTuzvGIwwu/bYfBl/lYn1BsFljrzxDIIsravjMoWs0Ky+18zM0qZpMNB/u/NiQZ6SeOij8AokdDpQiob/h
bp70qjlEjgfezJ3VOtjLs/GLblOU4C1bj@@##@@@#@@5XlBW/roNaMQg6bcpym+ufUiry6gIHy08d/Nqvec
dKTXY1fn5XPQb60ISOIf4UZd2Z9aYCnATyWxNglBtJF0TWYHCzRSUo0hJy1wFHXNxJnEa7PzVJzm+cb
wFmZktFrmlb9h3pKFhRI+xn1+I6XU+UDFMjXxw3EUGDxCJfDuMyZ9mr8AdRKq2JHzNkHEXh8+dQef6D
tBfp2980PLI0m50NTAlSZY5+z95RxZzU8dev8dWhIfZalLtuIPhTIbQ/Zv6V5XO29hEWATjXEsJ+OTX7vyfe
1eZMmOu/J97V5kyY678n3tXmTJjrvyfe1eZMmOu/J97V5kyY678n3tXmTJjrvyfe1eZMmOu/J97V5kyY678n3
tXmTJjrvyfe1eZMmOu/J97V5kyY678n3tXmTJjrvyfe1eZMmOu/5kyY678n3tXmTJjrvyfe1eZMmOu/J97V5ky
Y678n3tX97V5kyY678n3tXmTJjrvyfe1eZMmOgOH0r/1yaos=</EncryptedData>

V. RESULT ANALYSIS

Environment:

We have chosen 3DES and AES to encrypt values in our algorithm. To evaluate the structural XML encryption algorithm that is presented in this paper, documents of various sizes have been used on an Intel Core 2 Duo CPU P8400 @ 2.26 GHz, 4.0 GB RAM and the results shown in table 1 are obtained. The results obtained are due to the implementation of 3DES and AES.

Evaluation of Tag based XML Encryption	
Size of the Document	**Time to encrypt/decrypt (Milliseconds)**
1 KB	345,335
6 KB	355,345
10 KB	377,364
20 KB	384,392

Table 1: Evaluation of Tag based XML Encryption.

EVALUATION OF THE STRUCTURE MODIFICATION OF XML ENCRYPTION ALGORITHM	
Size of the Document	**Time to encrypt/decrypt (Milliseconds)**
1 KB	13,27
3 KB	35,82
4 KB	43,126
6 KB	82,189
10 KB	133,308
20 KB	256,596

Table 2: Evaluation of Structure modification XML Encryption Algorithm

The basic purpose of this paper is to introduce concept of structure modification encryption in XML file. The basic idea behind the methodology presented is that two keys are used for encrypting an XML document. One key is used for encrypting the content of xml element and the other key is used for encrypting the elements. Here element and content are separately encrypted so it will be difficult for attacker to decrypt it. Again symmetric keys are store on server in separate file which is also encrypted so without getting those keys almost it is difficult to decrypt xml file. Somehow if unauthorized users are able to get one key, even though they are not getting actual content of xml file. This method is also quite efficient as well as its performance is good.

We have evaluate single tag based encryption which is taken time according to Table-1 for different size. On the other hand in Table 2 we have written different values for our algorithm.

After all we have derive that time taken for encrypting and decrypting is also lesser then the single tag based encryption. This paper is little contribution toward security improvement of xml encryption.

References

[1] D. Eastlake, J. Reagle, D. Solo, F. Hirsch, and T. Roessler, "XML SignatureSyntax and Processing (Second Edition),"W3C Recommendation, 2008.
[2] D. Eastlake and J. Reagle, "XML Encryption Syntax and Processing," W3C Recommendation, 2002.
[3] XML Security by Claudio A. Ardagna, Ernesto Damiani, Sabrina De Capitani di Vimercati, and Pierangela Samarati.
[4] XML and Web Services Security Standards Nils Agne Nordbotten.
[5] http://www.w3.org/TR/xmlenc-core/#sec-Algorithms.
[6] A Stream-based Implementation of XML Encryption by Takeshi Imamura, Andy Clark, Hiroshi Maruyama.
[7] http://msdn.microsoft.com/en-us/library/windows/desktop/ms762271(v=vs.85).aspx
http://www.xml.com/pub/a/2001/08/08/xmldsig.html
[8] Understanding Web Services: XML, WSDL, SOAP, and UDDI. By Newcomer, E.