Abstract—A digital signature is an electronic signature that can be used to authenticate the identity of the sender of a message or the signer of a document. The implementation of digital signatures in XML documents has become a globally-recognized method for proving authenticity and providing data integrity and support for non-repudiation, to business messages for secure business transactions. XML based data is very useful for business transactions because of its various features like semantically rich and structured data, plain text format and Web-ready nature. This project mainly focuses on the ability of XML digital signatures to sign only specific parts of the document. In this paper we try to describe the features of an XML document and the XML digital signature standards in brief. Next we create a windows based application using Microsoft .NET cryptographic library in C# (programming language). The application lets a user encode and sign parts of a simple XML file. Finally we present three test cases in which the user can change the content of the message and the key and the application will give invalid authentication results.

Index terms---XML, XML, Digital signatures.

I. INTRODUCTION:

THE Digital Signature technology enables an enterprise to easily add electronic signing capabilities to electronic documents. Figure 1 below describes the basic mechanism of creating a digital signature.

The first step in the creation of a digital signature is to apply a hash function to the message, creating a message digest. The message digest is smaller than the original message and is fixed in length. To create the digital signature, message digest is encrypted with the sender’s key. The digital signature is usually included in the original message and sent.

For authenticating the receiver decrypts the message digest and compares it with the value obtained by hashing the original message. If the two values match the authenticity and integrity of the message is established.

Most business transactions today are done electronically and may involve participants who are physically present at many different locations. Thus a completed transaction may include information from various sources. It is therefore natural that a participant may wish to sign only that portion for which he or she is responsible. It is this context that makes the XML digital signatures so powerful. Older standards for digital signatures provide neither syntax for capturing this sort of high-granularity signature nor mechanisms for expressing which portion a principal wishes to sign.

Because of the basic nature of hash functions, the digital signatures only work if the verification calculations are performed on exactly the same data as the signing calculations. Even with an infinitesimally small change to the data, the message digest can change and thus the receiver may not be able to authenticate the user.

In some formats of data such as XML and HTML (Hypertext markup language) the spaces are ignored by the parsers i.e. if two XML documents contain the same nodes they will be considered equal, though their formatting may differ. Therefore if the representation of the signed data can change between signing and verification, then some way to
SI-3: Implementation of XML digital signatures

standardize the changeable aspect must be used before signing and verification. For example, even for simple ASCII text there are at least three widely used line ending sequences. If it is possible for signed text to be modified from one line ending convention to another, between the time of signing and signature verification, then the line endings need to be canonicalized to a standard form before signing and verification or the signatures will break.

II. XML DIGITAL SIGNATURES

Before going into the discussions of XML digital signature standards, we will present a brief overview of XML format.

XML document

XML is a markup language for documents containing structured information. XML was created mainly to facilitate the exchange of data over internet. It offers the flexibility of defining custom tags. An XML document has a logical tree structure. Each piece of data is represented as an Element. Elements are made up of an opening tag which contains the element name, the content and a closing tag which again contains the element name. Every XML document has to have what is called a ‘root element’. This is a pair of tags which enclose and describe the whole document. XML documents have one ‘root’ element in which other elements may appear, in which other elements may appear and so on.

Attributes are the properties of elements. While defining attributes, the attribute values must be surrounded by quotation marks but these can be either single or double. Entities allow us to insert information into a particular place, or various particular places, of an XML document. If we put an 'entity-reference' in a certain place, then when the XML file is being processed, it will replace that reference with the 'entity-content'. The entity-content could be a word or phrase, or even an entire XML document. Fig 2 below shows an XML document. This contains elements BLMNOS, CompanyA etc and attributes like namingSystem.

```
<?xml version="1.0" encoding="utf8"?>
<BLMNOS>
<CompanyA>
<DocumentInformation>
<documentName namingSystem="POSC">Sample 3</documentName>
<Security>
<securityClass>confidential</securityClass>
</Security>
</BusinessAssociate>
<AuthorizedPerson>
<name>Marvin P. Mooney</name>
<title>Vice President of Drilling Operations</title>
</AuthorizedPerson>
<Contact type="primary">
<name>Marvin P. Mooney</name>
<phoneNumber qualifier="office" type="voice">3071234567</phoneNumber>
</Contact>
</CompanyA>
</BLMNOS>
```

Figure 2. A sample XML document

III. XML DIGITAL SIGNATURES

XML digital signatures are the signatures available in XML format that serve as digital rubber stamps when appended to XML documents, ensuring the authenticity and originality of the parent document. XML digital signatures are very important in the web services based architecture of business processes.

Viewed from an n-tier application architecture perspective, the web service is a programmatic access to a piece of software application. The World Wide Web Consortium (W3C) defines a Web service as a software system designed to support interoperable machine-to-machine interaction over a network. Web services are application programming interfaces (API) that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services. Web services use open, XML-based standards and transport protocols to exchange data with calling clients.

```
<?xml version="1.0" encoding="utf8"?>
</BLMNOS>
```

Figure 3. Web service architecture
SI-3: Implementation of XML digital signatures

Non-repudiation and integrity assume alarming dimensions in this service-driven world. Suppose that a confidential XML document is populated by services A and B, and is passed on to service E by C and D. To make things worse, let's assume that these services are provided by different systems/vendors across the Net! Now, how can the end consumer service E ensure the authenticity of the sender services A and B? What guarantee does it have that the XML data passed on from A and B has not been modified during the transit or by other malicious services? With hacking becoming a common phenomenon on the Internet, it is quite possible that unauthorized services will come to life all of a sudden from the network and start consuming confidential business data. How can we make sure that the sender services A and B take full responsibility for what is contained in the XML document? It is in this context that the use of XML digital signatures becomes so important.

EXAMPLE

Figure 4. Simple XML based transaction

An example would be a real estate brokerage firm that might offer its contracts for offers on homes to be processed via web service as shown in figure 4. That way other agencies could partner with them and take advantage of the service and the processing needed for those types of deals. In our scenario, a real estate contract was sent, and it has multiple pages, each page needing a signature. As illustrated below, as the XML message was being read at the signer's end, the application would generate and track the appropriate Digital Signatures, ensuring the tracking, encryption, and time stamping of each signature. Upon completion of the message it would be returned to the sending business, where the signatures would be verified and logged. Alternatively, we could provide one Digital Signature for the entire document, a process that would be very similar to the one outlined above, only a bit quicker as there is less information to secure.

The XML digital signature technology is a combined effort between the W3C (World Wide Web Consortium) and the IETF (Internet Engineering Task Force). The XML Signature Working Group, which has members from both organizations, currently maintains the standards and specifications governing XML digital signatures.

IV. XML DIGITAL SIGNATURE ELEMENT

An XML signature can sign more than one type of resource. For example, a single XML signature might cover character-encoded data (HTML), binary-encoded data (a JPG), XML-encoded data, and a specific section of an XML file. Each of the data objects are first digested (a digest is fixed-length representation of a variable length data object and is created using an algorithm like SHA-1) and the resulting value is placed in an element (with other information). This element is then digested and cryptographically signed. Figure 5 shows a simple XML signature element. The signature element has the following structure (where "?" denotes zero or one occurrence, "+" denotes one or more occurrences, and "*" denotes zero or more occurrences). Signatures could be Enveloped or Detached. Enveloped signatures are performed over data within the same XML document as the signature, so a detached signature is over data that is external to the signature's element itself. Below we try to explain some important concepts related to these standards.

X-PATH

As explained above the XML Signature recommends a standard means for specifying information content to be digitally signed and for representing the resulting digital signatures in XML. The XML Signature specification meets this requirement with the XPath transform. XPath defines a way to compute a string-value for each type of node.

The primary syntactic construct in XPath is the expression. One important kind of expression is a location path. A location path selects a set of nodes relative to the context node. The result of evaluating this expression is the node-set containing the nodes selected by the location path. For example child::para selects the para element children of the context node whereas child::* selects all element children of the context node.

Canonicalization

As explained before this is important to take care of formatting differences in two logically similar documents. The canonicalization algorithm used normally is Base64. This transform requires an octet stream for input. If an XPath node-set (or sufficiently functional alternative) is given as input, then it is converted to an octet stream by performing operations logically equivalent to 1) applying an XPath transform with expression self::text(), then 2) taking the string-value of the node-set. Thus, if an XML element is identified by a barename XPointer in the Reference URI, and its content consists solely of base64 encoded character data, then this transform automatically strips away the start and end tags of the identified element and any of its descendant elements as well as any descendant comments and processing instructions. The output of this transform is an octet stream.
SI-3: Implementation of XML digital signatures

Types of XML digital Signatures:
XML digital signatures can be of three kinds
- Enveloped
- Enveloping
- Detached

**Enveloping Signature:**
The signature is over content found within an Object element of the signature itself. The Object (or its content) is identified via a Reference (via a URI fragment identifier or transform).

**Enveloped Signature:**
The signature is over the XML content that contains the signature as an element. The content provides the root XML document element. Obviously, enveloped signatures must take care not to include their own value in the calculation of the SignatureValue.

**Detached Signature:**
The signature is over content external to the Signature element, and can be identified via a URI or transform. Consequently, the signature is "detached" from the content it signs. This definition typically applies to separate data objects, but it also includes the instance where the Signature and data object reside within the same XML document but are sibling elements.

Next we try to summarize the steps needed in the creation of an XML document. This will also give a brief introduction of various elements.

1. **Steps to create XML Digital Signatures**
   1. Determine which resources are to be signed. This will take the form of identifying the resources through a Uniform Resource Identifier (URI). For example "http://www.abccompany.com/index.html" would reference an HTML page on the Web "http://www.abccompany.com/logo.gif" would reference a GIF image on the Web.
   2. Calculate the digest of each resource. In XML signatures, each referenced resource is specified through a <Reference> element and its digest (calculated on the identified resource and not the <Reference> element itself) is placed in a <DigestValue> child element like. The <DigestMethod> element identifies the algorithm used to calculate the digest.
   3. Collect the Reference elements.
      Collect the <Reference> elements (with their associated digests) within a <SignedInfo> element. The <CanonicalizationMethod> element indicates the algorithm was used to canonize the <SignedInfo> element. As explained before different data streams with the same XML information set may have different textual representations, e.g. differing as to white space. To help prevent inaccurate verification results, XML information sets must first be canonized before extracting their bit representation for signature processing. The <SignatureMethod> element identifies the algorithm used to produce the signature value.
   4. Signing.
      Calculate the digest of the <SignedInfo> element, sign that digest and put the signature value in a <SignatureValue> element. <SignatureValue>MC0E LE=</SignatureValue>
   5. Add key information.
      If keying information is to be included, place it in a <KeyInfo> element. Here the keying information contains the X.509 certificate for the sender, which would include the public key needed for signature verification.
   6. Enclose in a Signature element.
      Place the <SignedInfo>, <SignatureValue>, and <KeyInfo> elements into a <Signature> element. The <Signature> element comprises the XML signature.

2. **Steps to verify XML Digital Signatures**
   Following steps are needed to verify the signature element:
   Recalculate the digest of the <SignedInfo> element (using the digest algorithm specified in the <SignatureMethod> element) and use the public verification key to verify that the value of the <SignatureValue> element is correct for the digest of the <SignedInfo> element.
   If this step passes, recalculate the digests of the references contained within the <SignedInfo> element and compare them to the digest values expressed in each <Reference> element's corresponding <DigestValue> element.
V. .NET IMPLEMENTATION

The **System.Security.Cryptography** namespace in Microsoft .NET provides cryptographic services, including secure encoding and decoding of data, as well as many other operations, such as hashing, random number generation, and message authentication. The **sn.exe** utility in MS operating system provides key generation using machine as container.

The **System.Security.Cryptography** in .NET Framework implements an extensible pattern of derived class inheritance. The hierarchy is as follows:

- **Algorithm type class**, such as SymmetricAlgorithm or HashAlgorithm. These are abstract classes.
- **Algorithm class** that inherits from an algorithm type class; for example, RC2 or SHA1. These are again abstract classes.
- Implementation of an algorithm class that inherits from an algorithm class; for example, RC2CryptoServiceProvider or SHA1Managed. This level is fully implemented.

Using this pattern of derived classes, it is easy to add a new algorithm or a new implementation of an existing algorithm. For example, to create a new public-key algorithm, we can inherit from the asymmetric algorithm class. To create a new implementation of a specific algorithm, we will create a non-abstract derived class of that algorithm.

In our current implementation we have used symmetric algorithm DES. For a symmetric algorithm to be useful, the secret key must be known only to the sender and the receiver. In Microsoft .net RijndaelManaged, DESCryptoServiceProvider, RC2CryptoServiceProvider, and TripleDESCryptoServiceProvider classes are implementations of symmetric algorithms.

The above mentioned classes use cipher block chaining (CBC), and requires a key (Key) and an initialization vector (IV) to perform cryptographic transformations.

Project implementation:

In our C# project, we created wrapper classes like Encryptor, Decryptor, DecryptorTransformer, and Signer. As explained before **sn.exe** utility created the key pair. This key was stored in an XML file and then used again during decryption. Here is a brief summary of the steps involved in this project:

- Creation of an XML document.
- Creation of key pairs.
- Application of essential transforms to the content.
- Encryption and Signing of the content.
- Decoding of the XML document.
- Verification of XML signature.

Application Platform
- Microsoft.Net

Language used
- C#
SI-3: Implementation of XML digital signatures

Figure 7. Test case 1 (Part 1)

Step 4 - A character of the key is changed and again upon pressing verifies signature the document will not be verified. It will say invalid.

<RSAKeyValue><Modulus>76tM3ageQE/B9Z6m4yNhC1QygbznDnkUrz/CTiA18Kv+r+048+Zr96CmDTZOGExTgZZBWLsYzhEdq+ktAsoY93DdiPHCtL1160jKsJ/aQEP1AwILd0Nieahf1y8nEihHYAZGjuXZdsu5PRY0F3sS5jXvkkkn2eEgE=/</Modulus><Exponent>AQAs</Exponent></RSAKeyValue>

Figure 8. Test case 1 (Part 2)

Test Case 2

In this test case we are examining the verification of content. We take an XML file as input. It is encoded and decoded. In this time the key value is not changed but decoded message content is changed. Upon trying to verify we get “Invalid” result. Figure 9 demonstrates the results.

Figure 9. Test case 2

Test case 3

In this case we are examining the basic property of XML digital signature where individual parts of the document can be signed and verified. XML file shown below is taken as input. This has confidential information regarding two companies. We will encode and verify element of the xml document. This implies that the person sending the file is taking responsibility for partial information (Company A information) in the file. So in the test case we will sign one node (Company) of the document. It will be decoded and before verifying the signature we will change the information regarding company A. (We will change the name of the contact person). As soon as we will press verify signature, we will get “Invalid” result. However if we will change the information regarding company B we still will get “Valid” result.

As explained before regarding the X PATH transforms. We specify node Company A to be signed as follows: Reference.AddTransform(CreateXPathTransform("ancestor-or-self::CompanyA"))
SI-3: Implementation of XML digital signatures

<?xml version="1.0" encoding="utf8"?>
<BLMNOS>
<CompanyA>
<DocumentInformation>
<documentName namingSystem="POSC">Sample 3</documentName>
<Security>
<securityClass>confidential</securityClass>
</Security>
</DocumentInformation>
<BusinessAssociate>
<AuthorizedPerson>
<name>Marvin P. Mooney</name>
<title>Vice President of Drilling Operations</title>
</AuthorizedPerson>
<Contact type="primary">
<name>Marvin P. Mooney</name>
<phoneNumber qualifier="office" type="voice">3071234567</phoneNumber>
</Contact>
</BusinessAssociate>
</DocumentInformation>
</CompanyA>
<CompanyB>
<DocumentInformation>
<documentName namingSystem="POSC">Sample 3</documentName>
<Security>
<securityClass>confidential</securityClass>
</Security>
</DocumentInformation>
<BusinessAssociate>
<AuthorizedPerson>
<name>Marvin P. Mooney</name>
<title>Vice President of Drilling Operations</title>
</AuthorizedPerson>
<Contact type="primary">
<name>Marvin P. Mooney</name>
<phoneNumber qualifier="office" type="voice">3071234567</phoneNumber>
</Contact>
</BusinessAssociate>
</DocumentInformation>
</CompanyB>
</BLMNOS>

Figure 10. XML document for Test case 3

Figure 11 and 12 shows the results with company A information changed and company B information changed.

Figure 11. Test case 3 (Part 1)

Figure 12. Test case 3 (Part 2)
SI-3: Implementation of XML digital signatures

CONCLUSION:
We were able to use the Microsoft .Net Cryptographic libraries to verify the process and implementation of XML Digital signatures. The ease of use of using .net libraries is tremendous. We were also able to appreciate the utility of XML digital signature protocol in today’s web based business environment. This project can be further modified by implementing better user interface. The performance of various cryptographic service providers can be compared. We can further compare the .net implementation with java implementation of various algorithms.

Bibliography:

2. Extensible Markup Language (XML): http://www.w3.org/XML/
3. XML-Signature Syntax and Processing: http://www.w3.org/TR/xmldsig-core/
5. RSA Security: http://www.rsasecurity.com/
11. Enveloped Signature Transform: http://www.w3.org/2000/09/xmldsig#enveloped-signature