

# Securing transactions in OCEAN

*A summary of my approach so far...*

**Sahib Singh Wadhwa**

[ssingh@cise.ufl.edu](mailto:ssingh@cise.ufl.edu)

**Abstract:** *The primary purpose of using XML digital signatures for securing transactions, which otherwise can be handled by SSL, is to ensure non-repudiation. The secondary reason being the signing and encryption of selected elements of an XML document. The approach for developing XML digital signatures for signing contracts, and in future for the negotiations, is in accordance with the recommendations passed by the joint proceedings of W3C and IETF.*

## Introduction

In a scenario where a large number of computers are connected with one another there is a huge risk of security breaches. One of the most important of them is non-repudiation.

The transactions such as search request from the buyer, matching response from the sellers, agreement of contracts between a buyer and a seller etc., all use XML.

Apart from experiencing a number of benefits from the features of XML there are some very obvious disadvantages. Being text based it is vulnerable to attacks. This is when it comes the need to sign and encrypt XML documents.

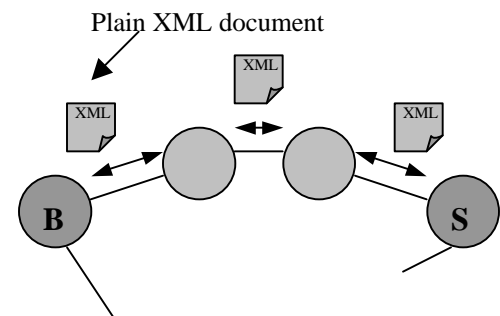
XML signatures provide application level security that conventional methods such as SSL (operating at transport-layer) do not. The most vital protection ensured by XML signatures is non-repudiation.

The challenge here is to guarantee that a digitally signed document has same legal status as a paper document with a handwritten signature. ABA has defined the requirements for a signed document to be considered legal. It is

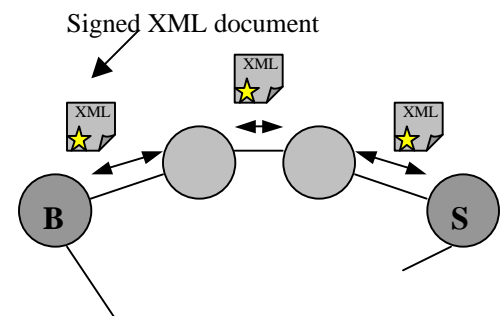
essential to be aware of such requirements when dealing with e-business/e-commerce systems.

## Goal

I intend to generate a prototype of an XML Signature module, which is able to demonstrate how a simple XML document (e.g. contract.xml, specifying terms of the contract) can be signed from one node, sent as a signed XML document over the media and verified on the other node.



**Figure 1:** Existing system



**Figure 2:** Proposed system

## Approach

A layer of XML signature module will be added to the existing layers of OCEAN. XML Signature layer will sit between matching/negotiation and lower layers. See figure 3.

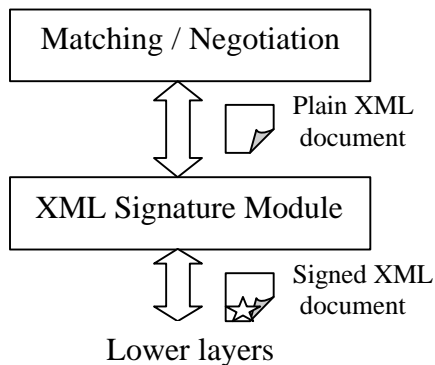


Figure 3:Node Structure

XML Signature has the following structure:

```
<Signature ID?>
  <SignedInfo>
    <CanonicalizationMethod/>
    <SignatureMethod/>
    (<Reference URI? >
      (<Transforms>)?
      <DigestMethod>
      <DigestValue>
    </Reference>)+
  </SignedInfo>
  <SignatureValue>
    (<KeyInfo>)?
    (<Object ID?>)*
</Signature>
```

## Signing an XML document

The process of signing an XML document is two-fold.

### Reference Generation

1. Determine which elements are to be signed.
2. Obtain transforms for each element (this is optional) in `<Transforms>` tag.

3. Calculate the digest of each element (using MD5 or SHA1 algorithm). This constitutes `<DigestMethod>` and `<DigestValue>` tags.
4. Put each reference element within a `<Reference>` tag.

### Signature Generation

1. Collect all reference elements in `<SignedInfo>` tag.
2. Apply Canonicalization algorithm. Include a `<CanonicalizationMethod>` tag.
3. Sign the resulting `<SignedInfo>` component with algorithm specified in the `<SignedInfo>`. Put the signature value in a `<SignatureValue>` tag.
4. Add key info (optional) in `<KeyInfo>` tag.
5. Enclose everything in a `<Signature>` tag<sup>1</sup>.

### Validating a signed XML document

Like Signing, the process of validating a signed XML document is also two-fold.

### Signature Validation

1. Verify signature value of the `<SignedInfo>` element. Recalculate digest of `<SignedInfo>` (using digest algorithm specified in `<SignatureMethod>`) and use public key to verify that the value of `<SignatureValue>` is correct for the digest of the `<SignedInfo>`. If it matches then proceed to reference validation.

### Reference Validation

1. Canonicalize the `<SignedInfo>` element based on the canonicalization method in the `<SignedInfo>`.
2. For each reference in the `<SignedInfo>`
  - i. Obtain the data object to be digested. (For example, the signature application may dereference the URI and execute Transforms provided by

<sup>1</sup> I have not decided yet which one of the three types of signatures (enveloped, enveloping and detached) to use for my implementation. I guess I will do a comparison and feasibility study to make a suitable choice.

- the signer in the reference element, or it may obtain the content through other means such as a local cache.)
- ii. Digest the resulting data object using the **DigestMethod** specified in its reference specification.
  - iii. Compare the generated digest value against **DigestValue** in the **SignedInfo** reference. If there is any mismatch, validation fails.

#### Key Validation takes place as follows:

1. Signed XML document is parsed by an XML parser, out of which key-info is separated from the signature.
2. Key info is fed to a trust engine which queries the root certification store for the certificates.
3. The certificate thus obtained is matched with the signature obtained from the original signed XML document.

#### Program Design

The design of classes is a direct result of the steps for signing and verification. The classes that I have designed so far are:

##### **class SecureTransaction**

This class is responsible for making a secure transaction over the existing plain XML transaction. It contains the following functions<sup>2</sup>:

```
SecureTransaction();
receivePlainXML();
generateXMLSignature();
sendSignedXML();
```

##### **class XMLSignatureGenerator**

This class is responsible for generating a signed XML document. It contains the following functions:

```
XMLSignatureGenerator();
getPrivateKey();
obtainRefElements();
canonicalize();
generateMessageDigest();
obtainTransforms();
sign();
generateSignedXML();
```

##### **class XMLSignatureValidator**

This class verifies the received XML document. The flow of functionality is almost the reverse of signing process. It contains the following functions:

```
XMLSignatureValidator();
reObtainRefElements();
reCanonicalize();
reGenerateMessageDigest();
reObtainTransforms();
getPublicKey();
reSign();
matchXMLSignature();
isMatch();
```

##### **class Register**

This class registers a user when it first registers with OCEAN. It obtains all the necessary information and generates a certificate. It contains the following functions:

```
getUserInfo();
generateXMLDocument();
sendRequestToCA();
receiveCertificate();
createLocalXMLCertificate();
```

##### **class CertificationAuthority**

This is a class that operates on the server side, probably in conjunction with the CAS. It is responsible for generating and managing certificates upon the request of CAS when any new user registers with OCEAN and when any node needs to verify the public key of any user

---

<sup>2</sup> function-signature and return-type is not shown to save space.

during transaction. It contains the following functions:

```
CertificationAuthority();
getRequest();
generateKeyPair();
generateCertificate();
signCertificate();
sendResponse();
```

## Conclusion

I hope if I am able to implement everything that I propose, this thesis can be very useful to OCEAN.

## References

- [1] Ed Simon, Paul Madsen, Carlisle Adams, "An Introduction to XML digital Signatures", August 2001
- [2] Murdoch Mactaggart, "An introduction to XML encryption and XML signature", September 2001
- [3] XML Signature WG, *The W3C and the IETF*, <http://www.w3.org/Signature/>
- [4] XML-Signature Syntax and Processing, <http://www.w3.org/TR/xmlsig-core/>, W3C Recommendation 12 February 2002
- [5] Canonical XML version 1.0, <http://www.w3.org/TR/xml-c14n>, W3C Recommendation 15 March 2001
- [6] XML Key Management Specification (XKMS2.0), <http://www.w3.org/TR/xkms2> W3C Working Draft, 18 March 2002
- [7] Toshiro Takase, Naohiko Uramoto, "XML Digital Signature System Independent of Existing Applications", IEEE Proceedings of the 2002 Symposium on Applications and the Internet (SAINT'02w)
- [8] Tatsuo Miyazawa, Takayuki Kushida, "An Advanced Internet XML/EDI Model Based on Secure XML Documents", 2000 IEEE
- [9] Karl Scheibelhofer, "Signing XML Documents", January 2001
- [10] Patrick W. Brown, "Digital Signatures: Are they Legal for Electronic Commerce", IEEE Communications Magazine 1994
- [11] Woo-Yong Han, Cheon-Shu Park, Shin-Young lim, Ji-Hoon Kang, "An XML digital signature for Internet e-business applications", Info-tech and Info-net, 2001. Proceedings. ICII 2001 - Beijing. 2001 International Conference