

Web Services Security

Scope for securing Web services – Different Standards & features

Vishnu Paturi G00508233
ISA 767 Secure E-Commerce
George Mason University
Fairfax, VA 22030
vpaturi@gmu.edu

Abstract:

With the growth of Internet, the business models and transactions got a new dimension to handle its business activities. This brought into the picture a new technology called “Web Services” which have become an integral part of Business to Business (B2B) interactions, and this B2B is an essential component of E-Commerce. Let it be the transfer of data between two different company server or providing service for other business (For ex: deals2buy.com etc), information exchange takes place using web services. The security of information in regard to web services is an important concern. This is because the main advantage of web services is simplicity and easily understandable (human readable) format of XML. Adding to that web services are sent over HTTP or other insecure medium where information tampering by eavesdropping, replay attacks, man in the middle attacks are possible. Even more is that Web Services have their own requirements for securing information which doesn't fit exactly into available technologies, this leads to the emergence of new standards promising to deliver different aspects of security in regard to Web Services.

In this paper we discuss different standards proposed to secure web services, what are the features each standard provides and for what application is that standard suitable for. This makes it easy for the reader to decide what standard to choose for his application development, what are the factors he should consider. Finally we discuss which standard scales well, what standard to use for a specific application and some conclude with some practices which increase productivity of application without leveraging performance.

1. Introduction

Modern businesses use a lot of IT infrastructure to carry out their business processes, manage resources and to communicate with its partners, suppliers and customers. In such a scenario, each key player in business (partners, suppliers and customers) needs to integrate their process and services with others such that everything works seamlessly and information is shared accurately within the time constraints. For implementing such a solution, we require what are called as middleware technologies that support custom built systems and integrate applications

to produce operational and management efficiency. Web Services are the emerging standard of middleware solution which uses SOAP (Service Oriented Architecture Protocol). With web service we can easily accessible services over a network regardless of its network structure or configuration, operating system, communication mechanism or implementing language. This is achieved through SOAP messages, which are composed of XML tags. XML is a language which defines data and also defines how to interpret the data using the metadata. By using such a middleware technology, a business model can save lot of time, money and IT infrastructure as one need waste time to discuss and decide on what software, platform and operating system should all partners and supplier use.

1.1 Need for securing Web Services

In order for communication to be successful, the communication system should provide confidentiality, integrity, non repudiation and performance (availability). Otherwise a communication system cannot be used to transferred sensitive data and it cannot be used for wide range of applications. In case of business communications, if the above mentioned properties are not provided, that communication system cannot be deployed.

So, web services need to be protected from attacks such as sniffing , spoofing, replaying, modifying intermediate packets etc and other attacks which enable theft of information. Eg: stealing creditcard numbers.

Another requirement is that , a particular class of services should be available to only partners and suppliers and customers shouldn't be able to use those services. Not only that some time a requirement may be in such a way that only a particular class of customers and normal customer shouldn't be able to use such services.

Consider an example where a company has two centers for operations providing different operations, these two centers provide information in the form of services such that only the company staff should be using the information although the information is transmitted over Internet. This is also a requirement which should be addressed by web services.

In order to address all these requirement, web services should have the feature of security, they should be able to authorize and reveal only necessary services to users.

1.3 SSL and its limitations

The main goal of designing SSL is to provide point-to-point security. But in Web services multiple intermediary nodes could exist between the two endpoints. In a typical Web services environment where XML-based business documents rout through multiple intermediary nodes, it proves difficult for those intermediary nodes to participate in security operations in an integrated fashion. This is because the application may be a mobile device based service where the infrastructure costs may shoot up, or it may be a legacy system which needs to be used. Thus we require end to end security in web services.

Second, SSL protocol provides security at transport level rather than at message level. As a result, messages are protected only while in transit on the wire. But consider the case where we need to store transmitted information in database so that it can be referenced later. (A typical e-commerce requirement)

Third, SSL cannot provide nonrepudiation of messages. But you know very well that nonrepudiation is very critical for business transactions. So we need some level of nonrepudiation for Web services-based transactions.

Most important is that SSL does not provide element-wise signing and encryption in a XML message. For example, if you want to encrypt only the creditcard field in a purchase order document and sign the whole transaction part of the document, it is not possible using SSL. This is because SSL is a transport-level security scheme.

2. Standard for securing Web Services

In this section we discuss different security initiatives for securing web services, why each one is important and significance of each model and how all of these work together are discussed in the sub sections below.

2.1 XML Digital Signatures

XML digital signature, is a digital signing technology, provides authentication, data integrity (tamper-proofing), and nonrepudiation. This project aims to represent digital signatures over any data type in XML syntax. The XML digital signature specification also defines procedures for computing and verifying such signatures. Another important area that XML digital signature addresses is the canonicalization of XML documents. Canonicalization enables the generation of the identical message digest and thus identical digital signatures for XML documents that are syntactically equivalent but different in appearance due to a different number of white spaces present in the documents.

XML digital signature provides a flexible means of signing and supports diverse sets of Internet transaction models. For example, you can sign individual items or multiple items of an XML document. The document you sign can be local or even a remote object, as long as those objects can be referenced through a URI (Uniform Resource Identifier). You can sign not only XML data, but also non-XML data. A signature can be either *enveloped* or *enveloping*, which means the signature can be either embedded in a document being signed or reside outside the document.

XML digital signature also allows multiple signing levels for the same content, thus allowing flexible signing semantics. For example, the same content can be semantically signed, cosigned, witnessed, and notarized by different people.

To Verify the signature, one has to trust the key sent or use PKI to retrieve the keys.

The picture below shows basic structure of the XML Digital Signature document.

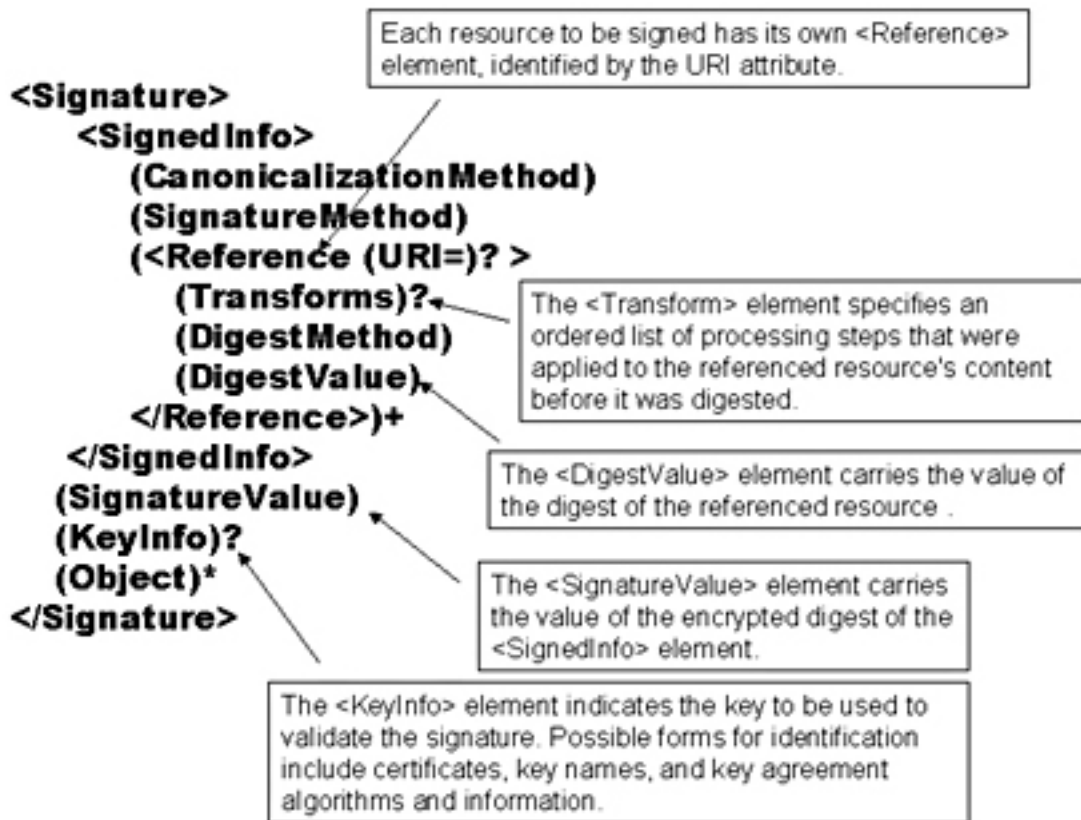


Image source : <http://www.xml.com/pub/a/2001/08/08/xmldsig.html>

2.2 XML Encryption

With XML Encryption, it is possible to encrypt selected fields of an XML document (credit card number etc) such it the date is transferred and stored securely, the other fields can be sent unencrypted. Each party can maintain secure or insecure states with any of the communicating parties. Both secure and non-secure data can be exchanged in the same document.

XML Encryption can handle both XML and non-XML (e.g. binary) data. The examples below demonstrate the use of Encryption.

Example 1:

```

<Invoice>
  <Order> <Item>Text book</Item> <Id>AFDC-2AQ3-1F3S</Id>
    <Quantity>2</Quantity>
  </Order>
  <Payment>
    <CardNo>393212-3210477-439850323</CardNo>
    <CardName>Mastercard</CardName>
    <ValidDate>06/02/2009</ValidDate>
  </Payment>
</Invoice>

```

In Example 2, the same XML document in Example 1 is used and only the Card Information is encrypted.

Example 2:

```
<Invoice>
  <Order>
    <Item>Text book</Item>
    <Id>AFDC-2AQ3-1F3S</Id>
    <Quantity>2</Quantity>
  </Order>
  <Payment>
    <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Content'>
    <EncryptionMethod Algorithm='http://w3.org/2001/04/xmlenc#tripledes-cbc' />
      <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
        <ds:KeyName>Vishnu Paturi</ds:KeyName>
      </ds:KeyInfo>
    <CipherData>
    <CipherValue>A23B45C564587</CipherValue>
    </CipherData>
  </EncryptedData>
</Payment>
</Invoice>
```

Now the the document can be sent over any medium and the receiver get the data. Once he receives this document the sender sends his key information using another XML message shown in Example 3 below.

Example 3:

```
<KeyExchangeMessage>
  <EncryptedKey CarriedKeyName="Vishnu Paturi"
  xmlns='http://www.w3.org/2001/04/xmlenc#'>
  <EncryptionMethod Algorithm= "http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
  <CipherData>
  <CipherValue>xyza21212sdfdsfs7989fsdbc</CipherValue>
  </CipherData>
  </EncryptedKey>
</KeyExchangeMessage>
```

This key is encrypted with the public key of the person to whom the encrypted message is sent.

We can use secret keys as well as public key cryptography for encryption. Everything is fine with this standard except that the exchange of keys and trust that the sender is sending the right key is still unsolved. If a PKI exists, this solves the problem.

2.3 XKMS (XML Key Management Specification)

XML Key Management Specification (XKMS) borrows the best of PKI without reducing scalability or security. XKMS creates a trust service that shields clients from complexity by providing an XML interface to PKI. One of the obstacles to PKI's wide adoption is that PKI operations such as public key validation, registration, recovery, and revocation are complex and require large amounts of computing resources, which prevents some applications and small devices such as cell phones from participating in PKI-based e-commerce or Web services transactions.

XKMS introduces an XKMS server to perform these PKI operations. In other words, applications and small devices, by sending XKMS messages over SOAP, can ask the XKMS server to perform the PKI operations. In this regard, the XKMS server provides trust services to its clients in the form of Web services and does not demand any change in their infrastructure on the users side. XKMS provides the ability to have a hierarchical key structure, and real-time analysis of the path through the hierarchy and makes it possible for parties to securely communicate without prior business arrangement.

A client and application server share an XKMS service on a known XKMS server to validate each other and to process requests between them. XKMS replaces many PKI protocols and data formats, such as Certificate Revocation Lists, Online Certificate Status Protocol, LDAP, Certificate Management Protocol and Simple Certificate Enrollment Protocol, with one XML-based protocol. XKMS also can be implemented client-to-client, server-to-client, server-to-server, and so forth.

The XKMS protocol provides three fundamental operations:

1. Locating keys and retrieving them so that clients can communicate securely with another clients/servers.
2. Validating, which makes sure the key is active and has not been revoked.
3. Registering services, which issues keys, reissues and revokes keys when necessary.

XKMS can works with the XML Digital Signature and Encryption standards. The server can respond with just the cryptographic key (for doing the signature math on small devices such as cell phones)

Many XML Web services standards, including SAML and WS-Security which are discussed in later sections use digital signatures to protect the content of authentication and message data. XKMS might be the specification that makes Web services implementation feasible in security perspective.

2.4 SAML (Security Assertion Markup Language)

SAML is an XML-based framework for exchanging security information. As a framework, it deals with three things.

1. It defines syntax and semantics of XML-encoded assertion messages.
2. It defines request and response protocols between requesting and asserting parties for exchanging security information.
3. It defines rules for using assertions with standard transport and message frameworks.

The main components of SAML include the following:

1. Assertions
SAML defines three kinds of assertions, which are declarations of one or more facts about a the user.
 1. Authentication assertions require that the user prove his identity.
 2. Attribute assertions contain specific details about the user, such as his credit limit or SSN.
 3. The authorization decision assertion identifies what the user is allowed to do.
2. Request/response protocol
This defines the way that SAML requests and receives assertions. Currently SAML supports SOAP over HTTP, but other protocols can also be added.
3. Bindings
This defines how SAML requests should map into transport protocols like HTTP.
4. Profiles
These dictate how SAML assertions can be embedded or transported between communicating systems.

While SAML makes assertions about credentials, it doesn't actually authenticate or authorize users. That's done by an authentication server. SAML does link back to the actual authentication and makes its assertion based on the results of that event.

SAML forms the basic backbone in Federated Identity systems.

2.5 WS – Security (Web Service Security)

The WS-Security specification defines a set of SOAP header extensions for end-to-end SOAP messaging security. It supports message integrity and confidentiality by allowing communicating partners to exchange signed and encrypted messages in a Web services environment.

WS-Security supports, integrates, and unifies several popular security models, mechanisms, and technologies. Since it is based on XML digital signature and XML Encryption standards, you can digitally sign and encrypt any combination of message parts. WS-Security supports multiple security models, such as username/password-based and certificate-based models.

It also supports multiple security technologies, including Kerberos, PKI, SAML, and so on. In addition, it supports multiple security tokens that contain Kerberos tickets, X.509 certificates, or SAML assertions.

WS-Security defines a standard extensions for all the messaging processes, transmission etc. These message headers can be used to implement integrity and confidentiality and non repudiation in Web Services applications.

WS-Policy defines the methods in which the capabilities and constraints of security policies can be expressed.

1. WS-Trust is a specification for establishing trust in between parties.
2. WS-Privacy is a specification that describes how privacy is stated and implemented by Web Services.
3. WS-Secure Conversation, describes how message exchanges can be securely transmitted, security context exchange and deriving session keys.
4. WS-Federation describes to transmit authentication information for supporting federated identity and single sign on for distributed computing.
5. WS-Authorization, is a standard for authorization data and policy management for Web Services.

WS-Security is a extensive sent of specifications of secure web services and a complete solution for secure web services.

2.6 ebXML (e business XML)

ebXML is a set of specifications that enable a modular electronic business framework for setting up business relations and carrying out business processes. The vision of ebXML is to enable a global electronic marketplace where enterprises of any size and in any geographical location can meet and conduct business with each other through the exchange of XML-based messages.

The main goal of designing this standard is to enable electronic business transactions to take place effectively. This standard proposes to create a central registry where

different business entities can discover each other and start a new business venture by agreeing on interfaces and standards which they will follow in future communications. This process is described in detailed below. To be convenient with the concept of ebXML, we need understand the meaning of certain terms. They are defined below:

Registry:

A central server that stores all the information about each party registered to the server and willing to do some business. The information a Registry stores and makes them available in XML are: Business Process of each party, Core Library, Collaboration Protocol Profiles of different parties registered, and Business Library.

Business Processes:

Activities that a entity registered in registry wishes to perform and for which it want one or more partners.

Collaboration Protocol Profile (CPP):

A profile filed with a Registry by a business entity wishing to engage in ebXML transactions.

Business Service Interface:

The Business Service Interface includes the specification of Business Messages the business supports and the protocols over which these messages might travel. (Similar to function signatures in programming language)

Business Messages:

The actual information communicated as part of a business transaction. A message will contain multiple layers. At the outside layer, an actual communication protocol must be used (such as HTTP or SMTP).

Core Library:

A set of standard functions which are required for every party and which are most commonly used are stored on the central registry server.

Collaboration Protocol Agreement (CPA):

In essence, a contract between two or more businesses that can be derived automatically from the Profile of the respective companies.

The figure below illustrates the ebXML messages which takes place in between companies which wish to interact:

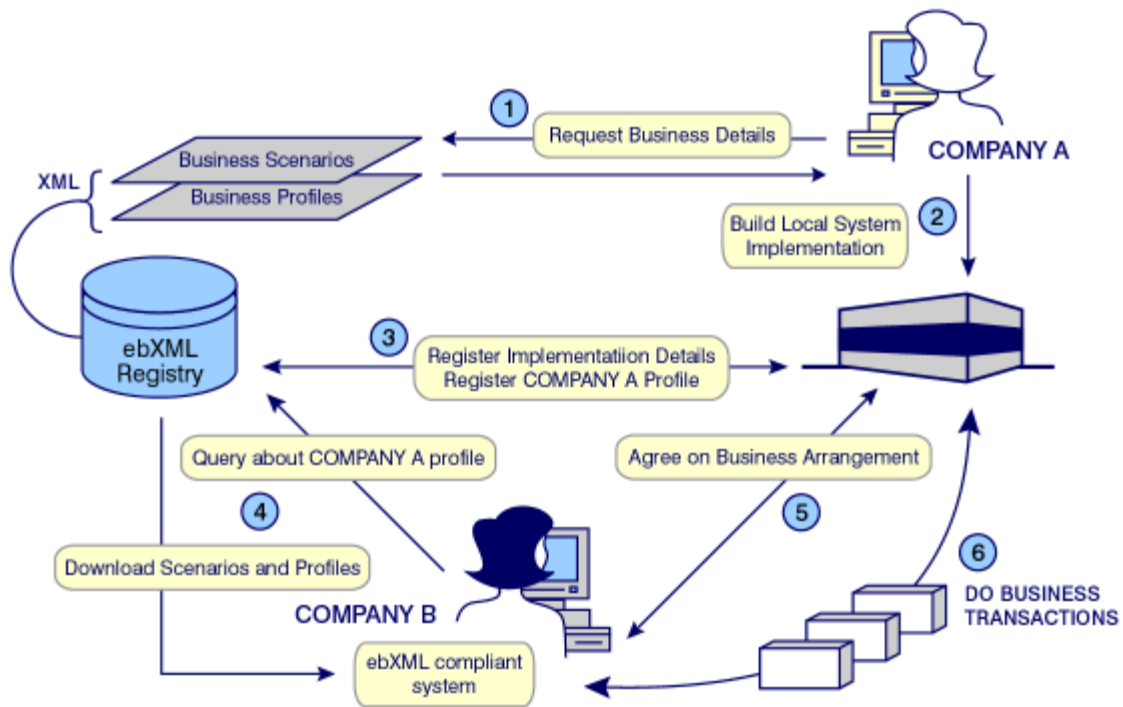


Image Source: <http://www-128.ibm.com/developerworks/xml/library/x-ebxml/>

In Step 1 of the figure, the company A which wishes to start a new business venture queries the ebXML registry to see if any business scenario matches its requirements, if a match is found, its profile is returned.

In Step 2, the company A implements a software system which supports and adheres to ebXML messaging service and also capable of handing the interface with Company B (the profile which matched its search on registry).

In Step 3, the Company A registers its profile on ebXML registry and its interfaces.

In Step 4, Company B queries the ebXML registry to see that Company B matches the business requirements and downloads its profile so that it can perform further communication with Company B.

In Step 5, either of the companies take the initiative of collaborating to do the business and in Step 6 business transactions are carried out according to the business interfaces defined in the registry.

3. Application domain and supporting technologies (Where to use what)

In this section we discuss how all the above standards work together to provide secure web services and where should we use which technology so that the application works at its optimum performance without compromising security.

- If one needs to just encrypt a message to secure against eavesdropping or sniffing of private information, use XML Encryption discussed in section 2.2 .
- If the requirement is to sign a message to provide integrity by preserving data in transmission and provide authentication of source, use XML Digital encryption.
- If both message confidentiality, integrity and source authentication is required, then first sign the message with XML digital signature and then encrypt it with XML Encryption.
- If trusted key management and trusted public key exchange is required, which is generally very essential for e commerce, use XKMS for exchanging keys, validation and certifying sources.
- If we want to have added security features for supporting applications in using federation and exchange of user identity, we use SAML which is built upon XML digital signatures and XML Encryption.
- If our application runs on good infrastructure and bandwidth (eg: a typical sweb based e commerce application.) use WS-Security.
- If one want to develop an application or a business model where business takes place in large scale in between partners, suppliers and buyers where people syndicate and use business strategies and deals. Use ebXML.

4. Best practices and techniques

In this section we discuss best practices for getting optimum performance out of the application in perspective of web services without having a trade off on security.

1. If you are designing web services for mobile computing (cell phones, pda, handheld, GPS etc) , just use that standard that meets the requirements. For eg: Don't use WS-Security just because it is a latest standard or because its being used by most developers currently. If you want just authentication use XML digital signatures, if you want encryption use XML Encryption, and as discussed earlier, use both if necessary. Even WS-Security uses the same for doing authentication and encryption. But if WS-Security is used in mobile computation, it may give a performance problems as the WS-Security framework is quit bulky when compared to XML encryption and signatures.
If source trust is necessary, use XKMS server rather than transferring certificates. This is because it avoids complex computations on client (mobile devices) and can boost performance considerably. (But most of them just use WS-Security and think it will take care of every security issue.)
2. To avoid delay in decryption of XML Encryption messages, encrypt only that part

of the message which is confidential and sensitive. If the session is long lived, exchange secret keys encrypted with the session key. This increases performance as secret key cryptography is much faster than public key cryptography.

3. In XML digital signatures, use the HMAC or MAC algorithm for more security.
4. Always remember, when we talk about security in web services, web services is a middleware technology. It just transmits data such that everything works seamlessly. As such these standards proposed doesn't implement security, they just transmit messages on both ends. Ultimate security lies in the cryptographic algorithm used and the key chosen. Don't think that if we use WS-Security with a small and easy key will ensure security.
5. If you are using PKI for validating certificates. Most or the big certification authorities (CA) like verisign, digicerts etc support have XKMS servers, if you are having an existing certificate, you have shift to XKMS by using their servers in your web service application.

5. Future of Web Service Security

The future of web service security standards are highly unpredictable. This is because web services are middleware which are used to integrate cross platform solutions. So if more number of companies and big shots like Microsoft and IBM support a particular technology, it may emerge as a future standard. May there are better standards proposed but not supported by these companies. So it all depends on business strategies and syndicates. But as of the present day, WS-Security is widely used and supported by many vendors across many platforms.

6. Conclusion

This paper aims in surveying various security standards proposed and understand what are the features provided by each standard and finally make the readers come to a conclusion as to what technology is more suitable for their web service application in terms of protecting security. I hope that the aim of the paper is met.

References

XML Digital Signatures:

<http://www.w3.org/Signature>

XML Encryption :

<http://www.w3.org/Encryption>

XKMS Uses public key for implementing Digital signatures:

<http://www.w3.org/TR/xkms/>

SAML:

XML-based framework for communicating user authentication, entitlement, and attribute information

<http://oasis-open.org/committees/security>

WS-Security:

Popular model of securing web services

<http://www.oasis-open.org/committees/wss/>

Microsoft standard for WS security:

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwssecur/html/securitywhitepaper.asp>

Securing Web Services using JAVA:

<http://java.sun.com/developer/technicalArticles/WebServices/security/>

Web Service Architect, More on WS-Security:

<http://www.webservicesarchitect.com/content/articles/apshankar04.asp>

Understanding Web Services:

http://.webopedia.com/DidYouKnow/Computer_Science/2005/web_services.asp

PKI and XKMS:

<http://www.networkworld.com/news/tech/2003/0908techupdate.html>

Security in a Web Services World: A Proposed Architecture and Road map

<http://www-128.ibm.com/developerworks/library/specification/ws-secmap/>

More on XML digital signatures:

<http://www.xml.com/pub/a/2001/08/08/xmldsig.html>