Implementation of Secured Web Service Model Using Stream Based XML Encryption and XML Signature

Srinath K.S.¹, Raj Sodhani², Shakthi Sharma³, Tripathi Putul Kumari⁴

solansrinath@gmail.com, raj23.sodhani@gmail.com, shakthisharma18@gmail.com, putulkumari10@gmail.com

¹,²,³,⁴ Sambhram Institute of Technology, Bangalore, Karnataka, India

Abstract

Being regarded as the new paradigm for internet communication, Web Services have introduced a large number of new technologies, such as Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI). These promote an environment for organizations to communicate in real time. The inevitable challenge being faced by these organizations today is to implement adequate Web Services Security as the Web Service transactions are done mainly through plain text XML (eXtensible Markup Language) formats like SOAP and WSDL, making them easy to get hacked.

This paper proposes an approach using XML Signature and XML Encryption over a Service Oriented Architecture (SOA) environment to ensure security to XML documents as well as to retain their structure. This framework facilitates security based on the parameters of authentication, integration and confidentiality.

Keywords: XML Encryption, XML Signature, Service Oriented Architecture (SOA), Simple Object Access Protocol (SOAP)

1. Introduction

Information technology has brought in a change in the way commerce is conducted around the world. The Web is different today than it was few years ago. Modern technology is setting up rules for how a company does business, how it communicates with customers and others in the new market places. It also has its impact on application-to-application communication. To improve the efficiency and flexibility of Web communication, Web Services technologies have been developed.

The concept of Web Services started in the late 1990’s and from then on has become the backbone of the IT industry. Currently all business transactions rely on Web Services to achieve their desired goals. Web Services and SOAs are often considered to be among the most important technological innovations of the last decade. A web service is any service that is available over the Internet, uses a standardized XML messaging
system, and is not tied to any one operating system or programming language [1][2].

This paper describes the techniques developed to secure these Web Services for an efficient, effective and secure communication. The techniques described include XML Encryption and XML Signature, which are implemented over SOAs using Web Services technologies such as SOAP and WSDL. Based on the design, analysis is done and results are obtained for various cases.

1.1 Motivation

In a web service, all the transactions take place in the form of XML Documents over the HTTP. The main disadvantage of this kind of file transfer is that XML is easily readable, which enables the attackers to easily get the information and misuse it for their malicious needs.

The ease with which attacks like eavesdropping and spoofing can be performed on XML Files, it requires certain amount of security for the data to be secured during transfer. If such security is not provided, then concepts like e-Commerce and online transactions would not be very popular because people would not find it to be trustworthy.

Without proper authentication provided to these XML Files, the users will not be able to identify the validity of an incoming document and cannot be assured if the data sent by them has reached the particular. This can lead to several problems where attackers can spoof a particular web service and cheat a customer by gaining access to their private information like credit card number and password.

To avoid the problems discussed above, techniques for providing security to the XML Files is needed, which can assure the integrity, authentication and confidentiality of the data. By providing this there can be an assurance that the transactions are safe from attackers and the users can be protected from them.

2. Web Services

Web services expose a standard interface that is platform and technology independent. By conforming to accepted industry-wide standards, Web services provide a means of communication among software applications running on different platforms and written in different application development languages and that present dynamic context-driven information to the user.

A web service has special behavioral characteristics:

2.1 XML-based

XML is used as the data representation layer for all web services protocols and technologies that are created, which can be interoperable at their core level. As a data transport, XML eliminates any networking, operating system, or platform binding that a protocol has.

2.2 Loosely coupled

A consumer of a web service is not tied to that web service directly; the web service interface can change over time without compromising the client's ability to interact with the service. A tightly coupled system implies that the client and server
logic are closely tied to one another, implying that if one interface changes, the other must also be updated. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.

2.3 Coarse-grained

Object-oriented technologies such as Java expose their services through individual methods. An individual method is too fine an operation to provide any useful capability at a corporate level. To build a Java program from scratch requires the creation of several fine-grained methods that are then composed into a coarse-grained service that is consumed by either a client or another service. The businesses and the interfaces exposed by them should be coarse-grained. Web services technology provide a natural to define coarse-grained services that access the right amount of business logic.

2.4 Ability to be synchronous or asynchronous

Synchronicity refers to the binding of the client to the execution of the service. In synchronous invocations, the client blocks and waits for the service to complete its operation before continuing. Asynchronous operations allow a client to invoke a service and then execute other functions. Asynchronous clients retrieve their result at a later point in time, while synchronous clients receive their result when the service has completed. Asynchronous capability is a key factor in enabling loosely coupled systems.

2.5 Supports Remote Procedure Calls (RPCs)

Web services allow clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support.

2.6 Supports document exchange

One of the key advantages of XML is its generic way of representing not only data, but also complex documents. These documents can be simple, such as when representing a current address, or they can be complex, representing an entire book. Web services support the transparent exchange of documents to facilitate business integration [2] [3].

3. Web Services Technologies

Several technologies have been introduced under the web service rubric and many more will be introduced in coming years. In fact, the web service paradigm has grown so quickly that several competing technologies are attempting to provide the same capability. However, the web service vision of seamless worldwide business integration will not be feasible unless the core technologies are supported by every major software company in the world. [4]

Over the past two years, three primary technologies have emerged as worldwide standards that make up the core of today's web services technology. These technologies are:

3.1 Simple Object Access Protocol (SOAP)
SOAP provides a standard packaging structure for transporting XML documents over a variety of standard Internet technologies, including SMTP (Simple Mail Transfer Protocol), HTTP (HyperText Transfer Protocol), and FTP (File Transfer Protocol). It also defines encoding and binding standards for encoding non-XML RPC invocations in XML for transport.

3.2 Web Service Description Language (WSDL)

WSDL is an XML technology that describes the interface of a web service in a standardized way. WSDL standardizes how a web service represents the input and output parameters of an invocation externally, the function's structure, the nature of the invocation (in only, in/out, etc.), and the service's protocol binding. WSDL allows disparate clients to automatically understand how to interact with a web service.

3.3 Universal Description, Discovery, and Integration (UDDI)

UDDI provides a worldwide registry of web services for advertisement, discovery, and integration purposes. Business analysts and technologists use UDDI to discover available web services by searching for names, identifiers, categories, or the specifications implemented by the web service. UDDI provides a structure for representing businesses, business relationships, web services, specification metadata, and web service access points.

4. Service-Oriented Architecture

Web services promote an environment for systems which are loosely coupled and interoperable. Many of the concepts for Web services come from a conceptual architecture called service-oriented architecture (SOA). A service-oriented architecture is the underlying structure supporting communications between services. SOA defines how two computing entities, such as programs, interact in such a way as to enable one entity to perform a unit of work on behalf of another entity. Service interactions are defined using a description language. Each interaction is self-contained and loosely coupled, so that each interaction is independent of any other interaction.

SOA configures entities (services, registries, contracts, and proxies) to maximize loose coupling and reuse. The "find, bind, and execute" paradigm as shown in the figure allows the consumer of a service to ask a third-party registry for the service that matches its criteria. If the registry has such a service, it gives the consumer a contract and an endpoint address for the service. SOA consists of the following six entities configured together to support the find, bind, and execute paradigm.

![Service Oriented Architecture](image)
4.1 Service Consumer

The service consumer is an application, service, or some other type of software module that requires a service. It is the entity that initiates the locating of the service in the registry, binding to the service over a transport, and executing the service function. The service consumer executes the service by sending it a request formatted according to the contract.

4.2 Service Provider

The service provider is the service, the network-addressable entity that accepts and executes requests from consumers. It can be a mainframe system, a component, or some other type of software system that executes the service request. The service provider publishes its contract in the registry for access by service consumers.

4.3 Service Registry

A service registry is a network-based directory that contains available services. This entity accepts and stores contracts from service providers and provides those contracts to interested service consumers.

4.4 Service Contract

A contract is a specification of the way a consumer of a service will interact with the provider of the service. It specifies the format of the request and response from the service. A service contract may require a set of pre conditions and post conditions. The pre conditions and post conditions specify the state that the service must be in to execute a particular function. The contract may also specify quality of service (QoS) levels. QoS levels are specifications for the nonfunctional aspects of the service. For instance, a quality of service attribute is the amount of time it takes to execute a service method.

4.5 Service Proxy

The service provider supplies a service proxy to the service consumer. The service consumer executes the request by calling an API function on the proxy. The service proxy finds a contract and a reference to the service provider in the registry. It then formats the request message and executes the request on behalf of the consumer.

4.6 Service Lease

The service lease, which the registry grants the service consumer, specifies the amount of time the contract is valid: only from the time the consumer requests it from the registry to the time specified by the lease. When the lease runs out, the consumer must request a new lease from the registry [5].

5. XML Encryption

Encryption is the method of conversion of the sensitive document into a form that is not understandable to unauthorized users. Authorized
users have to decrypt the ciphered text in order to understand the content. XML is a flexible way to create common information formats and share both the format and the data on the World Wide Web, intranets, and elsewhere. It is a formal recommendation from the World Wide Web Consortium (W3C), similar to the language of today’s web pages, the Hyper Text Markup Language (HTML).

The purpose of XML Encryption is to maintain the confidentiality of information during the encryption process with the help of Secure Socket Layer (SSL) or Transport Layer Security (TLS) or Virtual Private Network (VPN). The encryption process is controlled by an encryption key. In a more general approach, XML Encryption provides end-to-end security and is used to encrypt the XML document which further represents the encrypted data in XML documents. This is possible only when proper use of algorithms and technologies are defined. The reliability of the encryption algorithm depends on the size of the key or number of bits of that key [8] [9] [10].

6. XML Decryption

It is a process of decrypting the encrypted XML file. In simple terms, XML Decryption is used for the conversion of cipher text i.e. encrypted XML file to the original message i.e. plain text. This means that it is the reverse process of the XML encryption. During the process, the conversion of messages takes place by using decryption key. Only the correct decryption key results to obtain back the original message or file [11] [12].

7. XML Signature

XML Signatures also called as XMDS, XML-DSig, and XML-Sig are used for signing the digital content and verifying the digital signatures. This method is used to provide data integrity, so that no one can tamper with the information. XML Signatures are designed for secure transactions in XML format.

The XML Signature supports any type of digital signature encryption using all possible standard encryption algorithms. MD5, SHA-1 and RSA are some of the algorithms that are used to calculate the hash values of the data. The signature process is carried out on these hash values. After the hash value is signed, it is guaranteed that the integrity of the original document cannot be changed.

There are three basic types of XML Signatures: Enveloped Signature, Enveloping Signature and Detached Signature [13] [14] [15].

1. Enveloped Signature: Here, the XML Signature is included in the document itself and is the child element of the object being signed, as shown in the figure. The data being signed envelopes the <signature> and </signature> tags.

2. Enveloping Signature: Here, the document is included in the XML Signature
as the child element of the XML Signature. The data being signed is enclosed in the <signature> and </signature> tags.

3. Detached Signature: Here, the XML Signature is a separate document (mainly non-XML) from the signed XML document. The location of the signed document is given as a reference in the XML Signature.

8. Design and Implementation

This work is based on the various techniques to provide security to the data being transferred over the HTTP in the form of XML files. As transferring the XML files without any security makes it vulnerable to attacks such as Eavesdropping and Man in the middle attack, security measures are to be provided for data integrity, confidentiality and client/server authentication[6] [7].

8.1 Encryption and Encoding

The initial security provided in this design is the encryption and encoding of the XML File. This provides confidentiality to the data being transferred. To achieve this, we make use of the SAXParserFactory and DocumentHandler to parse the entire input file which is in the XML format.

It is read in the order of elements such as for example, <start> is the start element of any file and its corresponding end element is </start>. The entire data present in between these elements is considered as characters and is parsed accordingly. Each element is encrypted using DES Algorithm by providing a key, which is then encoded in the Base64 format as special characters can’t be transferred over the link in an XML File.

Each encrypted element is bound by <cipher> as their start element and </cipher> as their end element. The entire XML file is enveloped in <ra> and </ra> as their start and end document respectively. This file is transferred to the receiver end that decodes and decrypts it using the same key and algorithm, thus obtaining the file similar to the input file.

8.2 SOA Interaction

This section concentrates on the interactions and file transfer between the provider, registry and consumer of the Service Oriented Architecture. Here, the registry acts as the server and the provider and consumer both act as clients. The registry is divided into two servers, each for provider and the consumer.

The provider initially encrypts, encodes and places its WSDL file on the registry database. The registry stores it in the similar form and waits for its request from the consumer.
The consumer sends the file name to be required, and the server searches for it in its database. If available it sends that file to the consumer else it sends a “FILE NOT FOUND” message. After getting the file, the consumer decodes and decrypts the file using the same key and algorithm as that of the ones used in provider.

8.3 SOAP Messages

Once the WSDL interaction is complete, the provider and consumer interact directly using SOAP request and response. Both the provider and consumer act as client and server whenever needed.

In this design, the consumer initially encodes and encrypts a SOAP request being the client and sends it to the provider which acts as the server. After receiving the request, the provider decodes, decrypts the request and sends an encrypted and encoded SOAP response back to the client. Now, in this scenario, the provider acts as the client and the consumer acts like the server. The consumer, on receiving the response decodes and decrypts it.

8.4 Signature

Here, the messages to be sent between the sender and the receiver are signed to provide authentication and integrity. A third party acts as the signer and validator for these messages.

Initially the sender acts as a client and sends its data to be signed to the signer along with another file which contains a code verifying its authenticity. The signer compares the code with the code in its database, if found it uses DSA-SHA1 algorithm to generate a signed file where the signature is enveloped and passes this signed file back to the sender. If the code is not found, it does not sign the data and returns a message for the same to the sender.

Then, the sender sends the signed file to the receiver which is now acting as the server. The receiver stores the signed file in its database. It then passes this signed file to the validator for validating the received file. Here, receiver is the client and validator is the server, which validates and sends a text “SIGNATURE PASSED VALIDATION” in case there is no tampering with the data, else “SIGNATURE FAILED CORE VALIDATION” if any part of the data is modified in transaction.

Figure 8.1: System Architecture

9. Results

Based on the proposed design, results are obtained on various specifications of the design for particular cases. Initially, based on the size of the XML document, the time taken for it to be
encrypted and decrypted is analyzed and the results are given in Table 9.1.

Table 9.1: Encryption and Decryption Time

<table>
<thead>
<tr>
<th>No. of tags</th>
<th>Encryption time (ms)</th>
<th>Decryption time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>991</td>
<td>978</td>
</tr>
<tr>
<td>100</td>
<td>1115</td>
<td>1027</td>
</tr>
<tr>
<td>150</td>
<td>1227</td>
<td>1155</td>
</tr>
<tr>
<td>200</td>
<td>1506</td>
<td>1411</td>
</tr>
<tr>
<td>250</td>
<td>1517</td>
<td>1512</td>
</tr>
<tr>
<td>300</td>
<td>1765</td>
<td>1702</td>
</tr>
</tbody>
</table>

Figure 9.1: Graph on Encryption and Decryption Time

Figure 9.2: Graph on Key Sensitivity

The graph in Figure 9.2 shows the effect of change in the key on the output. Here, 0 means similar output, 0.5 means partial output and 1 means no output.

While performing signature on the input file, we also calculate the time taken for various sizes of input, the data is given in table 9.2. The graph in Figure 9.3 shows that as the size of the input increases, signature time increases.

Table 9.2: Signature and Validation Time

<table>
<thead>
<tr>
<th>No. of tags</th>
<th>Signature time (ms)</th>
<th>Validation time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>487</td>
<td>320</td>
</tr>
<tr>
<td>100</td>
<td>560</td>
<td>342</td>
</tr>
<tr>
<td>150</td>
<td>580</td>
<td>420</td>
</tr>
<tr>
<td>200</td>
<td>605</td>
<td>440</td>
</tr>
</tbody>
</table>

After this test, the modifications are done in the key while decryption. There are three outputs possible; either we get the similar output, partial output or no output. The modifications are done character wise starting with one, two and then three.
10. Conclusion

This paper provides the security measures like XML Encryption and XML Signature, which can be provided to overcome these issues. Using XML Encryption, security is provided from attacks like eavesdropping and using XML Signature, the data is secured against attacks like man in the middle attack. The analysis and results provide the necessary proof for the working of these measures.

This work can be enhanced to provide security measures like authentication, since the design does not keep a check on the user who is performing the transaction. The design can be upgraded to provide security against attacks like Denial of Service (DoS) and Replay attacks, as it is not keeping track of the number of times a user has accessed the system. This design can also be standardized to follow the WS-Policy.

References


Authors

Srinath K.S. B.E., M.Tech., MISTE. Lecturer, Department of Computer Science & Engineering, Sambhram Institute of Technology, Bangalore, India. His areas of interest are Web Service and Web Security.

Raj Sodhani, pursuing B.E., in Computer Science & Engineering from Sambhram Institute of Technology, Bangalore, India.

Shakthi Sharma, pursuing B.E., in Computer Science & Engineering from Sambhram Institute of Technology, Bangalore, India.

Tripathi Putul Kumari, pursuing B.E., in Computer Science & Engineering from Sambhram Institute of Technology, Bangalore, India.