Design Considerations for Implementing Security in Web Services

Fawaz Amin Alvi[1], Shakeel A Khoja[2], Zohra Jabeen[3]

[1] Sir Syed University of Engineering and Technology,
Department of Computer Engineering,
University Road, Karachi, Pakistan
alvi@iodomain.com.
[2] Karachi Institute of Information Technology
Gulshan-e-Maymar, Karachi, Pakistan
shakeel@kiit.edu.pk
[3] Sir Syed University of Engineering and Technology,
Department of Computer Engineering,
University Road, Karachi, Pakistan
zjabeen@ssuet.edu.pk

Abstract. The rapid growth in XML and increasing use of World Wide Web allow users to use internet as a platform for document sharing and hosting embedded with certain security features. XML is becoming most prevalent means through which documents and data are encoded for distribution among users on the web. Currently no strict security models and mechanisms are available that can provide specification and enforcement of security policies for XML documents. Such models are crucial in order to facilitate a secure dissemination of XML documents, containing information of different sensitivity levels, among (possibly large) user communities. This paper concentrates on proposing key design considerations to implement the PKI enables security in XML documents, by defining a component model and processing rules that can be shared across applications using common tools, avoiding the need for extensive customisation of applications to add security. The considerations reuses the concepts, algorithms and core technologies of legacy security systems while introducing changes necessary to support extensible integration with XML. This allows interoperability with a wide range of existing infrastructures and across deployments.

1.0 Introduction

There are two related forces that are transforming information technology today: the rapid growth of XML traffic on the network, and the widespread adoption of Web Services as a way of reducing the cost of integration and moving traditional enterprise architectures to flexible, Service-oriented architectures. Enterprises must plan ahead if they want to be able to manage the XML and Web Services on their networks [1]. Even more importantly, enterprises must take care to provide uninterrupted security for their IT environments. In the face of these changes, XML and Web Services introduce new security concerns for the IT manager, and new technology tools, including XML firewalls, offer the missing pieces of security that today's enterprises need.

Older security models, such as public/private key encryption model, do provide a set of core security algorithms and technologies that can be used as a wrapper over XML document, but don't allow working within the document itself and managing the contents. Along with, these standards were not designed to support common XML technical approaches for managing content, such as specifying content with uniform resource identifier strings or using other XML standard definitions for providing hyperlink services within XML content [2].

The best-known simplicity of XML is to provide portability of data between disparate business systems contrasts with the complexity of traditional and very structured Public-Key Infrastructure (PKI). A key architectural goal in the XML Key Management Specification (XKMS) is to shield XML application developers from the complexity of PKI implementation. It enables XML-based systems to rely on

complex trust relationships without the need for specialized end-entity PKI application logic on the client platforms where XML processing is taking place, thus reducing overheads.
To be able to understand the technologies behind web services security there is a necessity of explaining some fundamental definitions. This part will start with an explanation of security and than the rest of the paper covers the proposed framework for security techniques and technologies.

2.0 Traditional Methods of Security

There are various techniques that are being used to secure applications that are using XML as storage or communication medium. Few of the well known techniques are discussed as under

2.1 SSL / IPSEC

Since most of the XML based systems specifically web services doesn't have security solutions, it has had too rely on the security that other systems and layers provide. In result of that, SSL and IPSEC has become a popular way for patching the security lack in XML [3]. SSL uses handshake to authenticate a requester and responder with help of certificates. It also encrypts the exchanged data. SSL can only authenticate and encrypt a communication between two points. But requesting and responding to a XML based web service often takes more routes than just one. That would be an adequate solution if the route of the message is known in advance, but this is often not the case. Additionally, SSL only protects the communicated data during its transmission. Once the data arrives to the end point, the end host has to use other techniques to maintain the security.
Another problem with the SSL techniques for XML Web services is that SSL authentication and encryption consume a large amount of CPU time and consequently the transaction process is slowed down. Imagine how slow and laborious it would be for a server, to process several requests at the same time. A consequence of the above mentioned security lacks could is that exchanged messages get easier to copy.

2.2 HTTPR

HTTPR is a protocol for the reliable transport of messages from one application program to another over the Internet, even in the presence of failures either of the network or the agents on either end. Traditional HTTP is a very insecure protocol, for instance it never insurer's that the exchanged data get to its destination. In order to secure exchanging, the protocol has to implement SSL or rely on underlying protocols. Therefore an effort was made to create a new more secure version of HTTP, called HTTPR. HTTPR applies rules to make sure that all exchanged messages get to their target once and in the original form. If a message does not get to the target, HTTPR notifies the sender that the message was not received, and it a message sends twice, the second arrival will be discarded. HTTPR gives definitions to how to encapsulate a HTTP payload.

2.3 IP Blocking / IP Filtering

To limit a XML based application, there is a technique widely used by web community named IP blocking. In this technique when a user tries to connect to a web service, whether the user has the right to

the XML based web-service / application is checked by looking its IP address. The web service provider has to maintain a list with IP addresses, from which requests are valid. The provider compares the users IP address with the list. But IP blocking blocks access from all IP address expect those in the list, this means that not only the web service will be blocked from the user, also the WSDL, which potential customers can view for further information about the web service. An additional problem with this technique is that it requires administration of the Access IP list. If a specific web service is widely used, the IP list will grow large and more difficult to maintain. This also requires that the list is updated regularly to avoid revoked accesses. Also an unauthorized user can access to web service using IP spoofing.

3.0 Native XML Security Techniques

As XML usage became widely popular among web developers, the application architects start thinking of specialized techniques for implementing security inside XML documents. These were not special purpose encryption/decryption techniques designed specifically for XML but in real essence steps and ways to help code developers to use traditional crypto graphical techniques to be used with XML. Three really powerful candidates for native XML security techniques are:

- XML Signatures Specification
- XML Encryption Specification
- XML Key Management Specification

With the help of XML signature, only desired parts of an XML document can be signed and verified. Using XML encryption asymmetrically / symmetrically encryption of specific parts of an XML document is possible. E.g. to encrypt a specific element of a XML Document
XML encryption and signature tackles three security issues, these are authentication, message integrity, and non-repudiation. One can wonder why ordinary signature and encryption cannot be used directly, the answer is that, XML encryption and signatures, in contrary to the ordinary techniques, can encrypt or sign specific parts [4]. This functionality is important because in some cases, some XML needs to be protected and other parts needs to be available. An unprotected part can be those which may be changed.

4.0 Native Web Services Security Techniques

As the use of web services are increasing, various companies are providing their own libraries and toolkits for creating web services with specialized features and security techniques. These techniques basically make use of native xml security techniques but provide wrappers over existing ones. These specialized techniques are basically the result of extensive research done by individuals or organizations and can be termed as Native Web Services Security Techniques.

4.1 Soap Sec

SOAP, the Simple Object Access Protocol, is XML syntax for exchanging messages. Because it is XML, it is both language and platform independent. SOAP uses the same port as HTTP, and is therefore a common used protocol for exchanging data between networks. But SOAP was never developed to

consider security issues; instead it relied on the underlying protocols security handling such as HTTP, TCP/IP etc [5]. Since SOAP messages consists of XML code, anyone who sniff's up the messages can se the exchanged data. Important exchanged data can be passwords etc. Therefore it would be adequate for that information to be encrypted. Another problem with SOAP messages is that they are one way transmissions; this gives the consequence it gets easier to steal, sniff, and resent the messages. In order to overbuild some of the security problems in SOAP, a new technique was proposed by IBM and Microsoft, called SOAP-sec. SOAP-sec enables message signing by using an added header to SOAP.

4.2 WS Security

Ws security specification was worked on by some major companies such as IBM, Microsoft and VeriSign. Ws security was formed to overbuild the lack of security that exists in Web services and also to provide a standard for secure message exchange, signing etc. WS security does not solve all the security problems and nor does it give a specific model on how web services should be built. It only gives guidelines. But it is important to notice that is just one specification in a row of others. The specifications focus is on SOAP extensions.

The extensions provide authentication, message integrity confidentiality and signature to messages. The general guidelines in WS security are focused on Authentication and Encryption. WS security does not limit itself to a specific model or mechanism; on the contrary it has support for several models and security mechanisms. For instance, a developer can use software tokens as well as hardware tokens. WS security has several requirements on system [6]. These specifications are WS-policy, WS-trust, WS-privacy, WS- Secure Conversation, WS-Federation and WS-Authorization. All these specifications together aims to tackle all the known existing security problems in Web services.

4.3 Design Consideration during Implementation

Security of a web service should be designed with a goal to fit together the ideas of the traditional security techniques in the light on the specifications as provided by W3C for XML Security and Web Security. The result will be a standard and high performance implementation. There are many technical reasons why various available XML based security providing standards could not be used together separately. All of the guidelines available on the web related to applying security are relatively new and are made specifically to focus on some specific aspect.

Not a lot of work is being done on the integration of such aspects and developers find it difficult to incorporate security inside their service oriented architecture despite of having very robust set of traditional available security tools. The major goal is to provide an easy way implement security in web services in integration with traditional cryptographic techniques [1, 7].

The following information is intended to introduce security in web services or service oriented architecture. The following model make use of available native XML technologies as discussed earlier but gives a consistent singular API/Design to developers to introduce security into their applications but the essential requirement is to produce methods that will work naturally with contents created using XML and to guarantee the aspects of integrity, confidentiality, authentication and accountability.

5 Component Oriented Design

Developers should define the security system in forms of components. It can consist of various independent components. These components should be independent and will be providing atomic operations that are needed while implementing a secure a web service. Each component should be designed with respect to the level they are categorized into. For example level 2 components will be using the functionalities of level 1 components so cannot be designed before level 1 components.

The Level 1 Components constitutes the first layer of the model. It consist of the all the basic level functionality needed. The level 1 component will give an interface to the next level components to use the traditionally available methods for security and also provide XML parsing capabilities to the other level components. First level components are:

- XML Parser Components
- Traditional Security Components
- Request / Response management Component
- Inter-Components Communication Component
- Key Management Components
- XML Transformations Component

The level 2 components are basically the core of the security model. These components will be providing the methods and their implementations that are defined or outlined in the various XML / Web Services Security related specifications provided by the W3C. There is one very important consideration of these components that is they will be talking XML as input and after successful operation the output will also be an XML document. Thus a true XML developer will be enjoying the fun of using XML and also making the web services conforming to one of property of XML documents i.e. XML in and XML out. All the information is in XML format.

- XML Key Management Component
- XML Encryption / Decryption Component
- XML Signature / Validation Component
- XML Access Control Component

The level 3 components are basically the components designed for web services layer. They will be making use of level 2 components for processing raw XML but in the output giving implementation dependent output for particular web services. These components will be use acting as a web services security library. These components are:

- SOAP Security Components
- XML RPC Security Components
- WS Security Components

5.1 Logical View of Web Service Security Model

There will be a two way flow of XML data inside the model. One way the XML data will move to get secure and as it moves towards the end the relevant security assertions are inserted over it and when it gets out, the resultant XML will be a perfect secure XML following all the standard XML security standards. The other way in, will be the secure XML document and its output will be the XML that

would be extracted as the result of applying the various XML security assertions or access control policies.

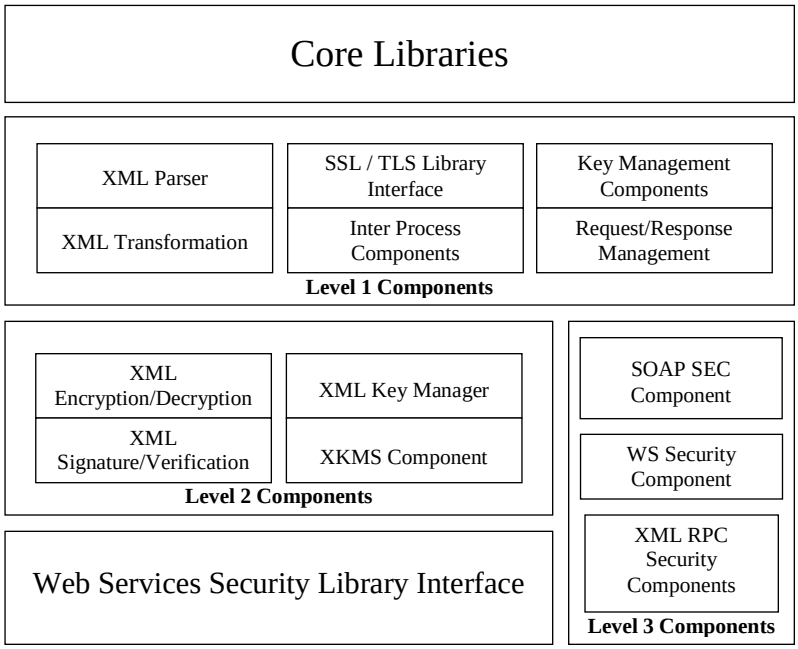

Figure 1: *A Basic Web Services Security Model, showing various component levels.*

5.1.1 XML Signature / Encryption Component

In the Encryption and Signing process, there can be a possible need of counter signature, or partial encryption, it is therefore better to perform signature or encryption by processing input XML along with a stencil/template that specifies a signature or encryption skeleton, the way to use transformation component, the usage methods for traditional cryptographic/algorithms components, and the way to interface with XML key management component for key selection process.
This stencil document will be an XML document itself with same in structure as the desired result but some of the nodes will be left empty and will be filled by the XML encryption/signature components after performing relevant computations.
The key for signature/encryption can be obtained from the key managers in the key management component using the information from the stencil document, does necessary computations and puts the results in empty nodes of the given stencil. Signature or encryption component controls the whole process and stores the required temporary data. Since the Stencil information is also a XML file, it might be created in advance and saved in a file and can be given to the application as an input otherwise the security API will have to generate a stencil by itself gathering information by itself.

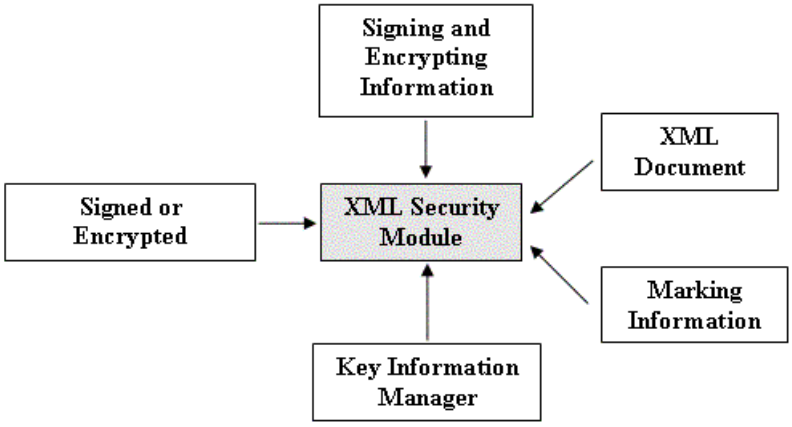

Figure 2: *flow diagram of XML Encryption / Signature Components, taking plain XML document along with marking information, by using the KeyInfo from Key Management Components.*

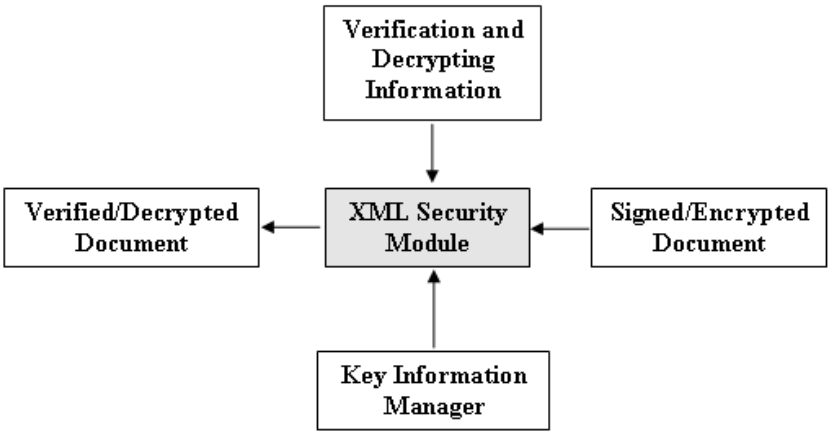

Figure 3: *Flow diagram of XML Decryption/Verification Components, taking signed document as input, extracting the algorithm information from the document and using the appropriate key from key manager decrypts or verifying the document and sending the result as output.*

This logic allows application also to be able to generate stencils without depending on functions provided by the component model. Also in some cases stencil should be inserted in the signed or encrypted data *(for example, to create an enveloped or enveloping signature).* Signature verification and data decryption do not require template because all the necessary information is provided in the signed or encrypted document.

5.4.2 Transformation Components

Since XML Digital Signature and XML Encryption standards are very flexible and provide an XML developer many different ways to sign or encrypt any part or even parts of an XML document [6,7]. The key for such great flexibility is the transforms model defined by these specifications. Specifications define transform as a method of pre-processing binary or XML data before digest or signature calculation/verification. To extend this definition, a name "transform" can be used for any operation performed on the data: reading data from an URI, XML parsing, XML transformation, calculation digest, encrypting or decrypting. Each transform provides at least one of the following call-backs: "push binary", "push XML", "pop binary" or "pop XML".

In order to simplify transforms development, additional "execute" call-back can be added. This call-back updates internal transform buffers and is used by the "default" XML/binary push and pop call-backs. For example, most of the crypto transforms could be implemented by just implementing one "execute" call-back. However, in some cases using push/pop call-backs is more efficient.
When necessary, a transforms chain can be constructed as specified in the template or document and processes data by "pushing" or "popping" through the chain. For example, then binary data chunk is pushed through a binary-to-binary transform, it processes this chunk and pushes the result to the next transform in the chain. The following transforms chain might be constructed during digest calculation.
It should be made sure that output data type (binary or XML) of previous transform matches the input data type of the next transform by inserting XML parser or default C14N when necessary [8]. Custom transforms could be added by the crypto plug-in or application at any time.

5.4.3 Key Management Component

Processing some of the key data objects require additional information which is global across the application. For example, X509 certificates processing require a common list of trusted certificates to be available [9]. All the common information for key data processing can be kept in a collection of key data stores called "keys manager". Keys manager can have a special "keys store" which lists the keys known to the application. This "keys store" can be used to lookup keys by name, type and crypto algorithm (for example, during <dsig: KeyName/> processing). Simply a flat list based implementation of a key store can be used. However the web service can replace it with its own or any other keys store (for example, based on an SQL database). Keys manager is the only of component in Level 1 of the XML Security Component Model which is supposed to be shared by many different components. Usually keys manager is initialised once at the application start-up and later is used by routines in "read-only" mode. If application or crypto functions need to modify any of the key data stores inside keys manager then proper synchronization must be implemented. In the same time, application can create a new keys manager each time it needs to perform XML signature, verification, encryption or decryption.

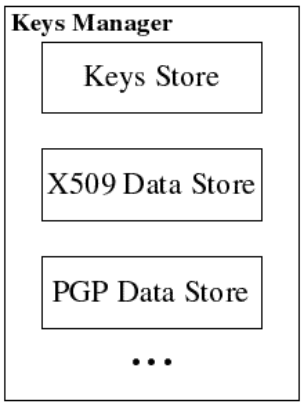

Figure 2: *Key Manager*

5.4.4 Inter-Component Communication Components

Application may control various components behaviours using several context objects. New context objects are created for each operation and could not be reused. Context objects can be used to store temporary data and return additional information to the application.

6 SUMMARY

These are the considerations that can be helpful for developers while implementing security into web services. But a lot of issues will come during the exact implementation. These considerations were found during the implementation of security for a web service. There is still a lot that can be done on this area. The features that are lacking to be discussed in this paper are: Access Control Component and Digital Rights Protection Component. These modules are not used in most of the XML applications but are part of security of any system. In-fact these components are not rated as very necessary.
This paper is not the exact solution to the problems of XML/Web Services security, but it is basically to show the way of how all the various available methods of implementing security can fit together to provide what is actually needed. In Future it will improve and new consideration will be found as implementation will be done completely and as maturity level increase it may one day can act as the exact solution.

References

1. Menezes, P. van Oorschot, and S. Vanstone (1996), “Handbook of Applied Cryptography” CRC Press USA, 1996.
2. Blake Dournee (2002): “XML Security” McGraw Hill Inc USA, 2002
3. Jake Sturm (2002): “Developing XML Solutions”, Microsoft Press Inc, USA 2002
4. R Allen Wyke, Sultan Rehman & John Brad (2001): “XML Programming”, Microsoft Press Inc, USA, 2001

5. E. Bertino & E. Ferrari (2002), “Secure and Selective Dissemination of XML Documents”: ACM Transaction on Information and System Security, Vol 5 No. 3 August 2002
6. Deutsch, A., Fernandez, M., Florescu, D., Levy, A., And Suciu, D. (1999), “Securing XML documents” Proceedings of the International Conference on World Wide Web, W3C.
7. XML Encryption” W3C XML Encryption Working Group, http://www.w3c.org/Encryption.html
8. “XML Digital Signature” W3C XML Digital Signature Working Group, http://www.w3c.org/Signature.html
9. XKMS” W3C XKMS Working Group, http://www.w3.org/2001/XKMS