Developing a Framework to Implement Public Key Infrastructure Enabled Security in XML Documents

Fawaz Alvi
Graduate Student
SSUET, Karachi

Shakeel A. Khoja
Assistant Professor
KIIT, Karachi

Zohra Jabeen
Graduate Student
SSUET, Karachi

Abstract: *This paper concentrates on proposing a framework to implement the PKI enables security in XML documents, by defining a common framework and processing rules that can be shared across applications using common tools, avoiding the need for extensive customization of applications to add security. The Framework 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. 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.*

Keywords: *XML, Web Services, Security, Public Key Infrastructure*

1.0 Introduction

Recent advancement in XML and increasing use of World Wide Web allow users to use internet as a document sharing and hosting system, with certain security features. XML is becoming most prevalent means through which documents and data are encoded for distribution among users on the web. At the network and document security front, 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 [1].

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 technology behind XML security there is a necessity of explaining some fundamental definitions. The second section of this paper defines some explanations of security and then the rest of the sections define the proposed framework to implement security in XML documents.

2.0 Traditional techniques used for security of XML applications

There are various techniques that are being used to secure applications that are using XML as storage or communication medium. All these techniques are wrapped over a XML document to provide security features and none of them can be embedded within the document.

Securing a XML document with SSL/IPsec is the common way to provide security [2]. 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. Requesting and responding to a XML based web service often takes more routes than just one, providing 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.

For example if to points (A and D) with two intermediate points (B and C) want to communicate with each other, A has to go throw B and C in order to get to D. In this case it cannot use SSL to authenticate itself and D, since SSL can only authenticate between two points. It could use encryption, but this requires intermediates B and C to be able to encrypt and decrypt the messages in order to be able to process them. 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.

IPsec is a secure format of IP. It consists of two parts, AH (Authentication header) for authentication and ESP (Encapsulating security payload) for encryption. IPsec is intended to provide authentication, integrity and confidentiality. This technique provides message integrity, signature and encryption of exchanged data, but lacks videlicet point to point authentication and the possibility to crypt and sign selective parts [2,3].

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 insures the exchanged data to reach 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 [5]. 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 if a message is received more then once, only the first instance is retained. HTTPR provides definitions about how to encapsulate a HTTP payload. To achieve the same benefits as HTTPS, there is a mixed version of HTTPR and HTTPS called, HTTPSR.

IP blocking is another method to provide security. The users and their rights are restricted by checking the IP address, where they are coming from. The web service provider can maintain a list with IP addresses, from which requests are valid. The provider compares the users IP address with the list. A problem occurs with this technique when the IP list grows phenomenally huge and becomes very difficult to maintain. Sometimes unauthorized access can also be obtained through IP spoofing.

With the help of XML signatures, one can sign only desired parts of an XML document. With the help of XML encryptions, one can also asymmetrically/symmetrically encrypt parts of an XML document. XML signatures become useful in the cases where a part of system needs to be protected and other parts are kept open to public.

XML signature contains information about used signature techniques such as hashing and encryption algorithms. With this technique, assurance is granted that data has not been changed. XML Signatures differs from ordinary checksums, through association between the signature and the key. XML Signatures can be done in three ways i.e.

- to reside the signed data inside the signature
- opposite to the above, the signature resides inside the signed data
- Both the signed data and signature resides a different XML file [4].

3.0 XML encryption techniques

XML encryption ensures that data cannot be read by unauthorized users. Unlike SSL, this technique enables the data to be protected both in transport and on the user's computers. This specification gives guidelines about algorithms, key information etc. XML encryption can use both symmetric and asymmetric keys, but more widely symmetric since these techniques are less CPU intensive and are therefore more suitable for larger data exchange.

XML Key Management Specification (XKMS)

To be able to use PKI, a system that can handle certificates and keys must exist. XML version of PKI handling is called XKMS (XML Key Management Specification). XKMS gives guidelines on how to integrate keys, certificates with applications and guidelines about registration, revocation and updates. XKMS uses SOAP over an HTTP based network. XKMS consists of three parts, i.e. X-KISS (XML Key Information Service Specification), X-KRSS (XML Key Registration Service Specification) and Protocol binding specification [5,9].

In order to achieve high security these standards need to be used in conjunction. An XML encryption must be able to use signatures or else the sender cannot be trusted and the XML signature has to use XKMS to handle keys that are exchanged. An important issue to be considered concerning XML encryption and signatures is that, these techniques are new and there is not an implemented prototype using these techniques.

SAML and XACML

SAML (Security Assertion Markup Language) is a standard that gives recommendation about how security information should be exchanged, using Internet. SAML gives guidelines on assertions to request and response messages in order to provide authentication and authorization. SAML shows how single sign on can be achieved when several web services are interacting. This means that a web service does not need to authenticate itself every time it needs a further web service, it can authenticate itself towards a trusted web service and delegate that it is authenticated when requesting for usage of other Web services. SAML can provide these adding XML assertions. SAML, like vice Ws security, uses techniques such as XML signature and encryptions.

XACML is a set of rules of how authorization over the internet should take place. XACML defines the representation for rules that specify the who, what, when and how of information access. XACML can be considered as a complementary to SAML. When a web service finds a SAML element in the XML document, it processes the request by checking its XACML policy through PRP (Policy Retrieval PDP).

SOAP-Sec

SOAP, the Simple Object Access Protocol, is XML syntax for exchanging messages. SOAP 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. Since SOAP messages consists of XML code, anyone who sniff's up the messages can see the exchanged data. A major problem with SOAP messages is that they are one way transmissions giving 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 [5,10].

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. Although, WS security has several requirements on system, such as:

- The Web service language must support multiple security tokens
- Several cryptography technologies
- Sender to requester security
- Transport security [6].

4.0 Designing of a Framework to secure XML

An essential requirement of new security framework is that it should work naturally with content created using XML. The overall objective of designing a XML Security Framework is to guarantee the aspects of integrity, confidentiality, authentication and accountability (key management). These aspects shall be taken as high level requirements to the design of framework and should be further elaborated to arrive at more tangible requirements.

The XML Security Framework is designed with a goal to fit together the ideas of the XML Encryption and XML Digital Signature in the light on the specifications as provided by W3C into one 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 in XML Documents 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 XML based setups despite of having very robust set of traditional available security tools. The major goal of the XML Security Framework is to provide an easy way implement security in any XML based application in integration with traditional cryptographic techniques.

XML Security Framework is intended to introduce security in XML applications or web services. The framework 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.

4.1 Components of Framework

The Framework defines the security system in forms of components. These components will be independent and will be providing atomic operations that are needed while implementing a secure XML based application. The components should be designed with respect to the level they are categorized into, such as level 1 and level 2 components.

At the first layer are level 1 components, consisting of all the basic level functionality that is required by any security systems. The level 1 components 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 higher level components, such as parsers, request / response management components and inter-communication components.

The level 2 components are XML transformations components, key management components, encryption/decryption components, signature/validation components and access control components, forming the core of the framework. These components will be providing the methods and their implementations that are defined or outlined in the various XML 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 framework conforming to one of property of XML documents i.e. XML in and XML out. All the information is in XML format.

There will be a two way flow of XML data inside the framework. 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. A simple flow diagram is shown in figure 1.

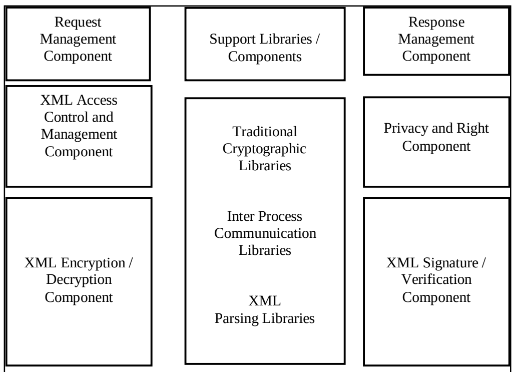
Request Management Component
Support Libraries / Components
Response Management Component
XML Access Control and Management Component
Traditional Cryptographic Libraries
Privacy and Right Component
Inter Process Communuication Libraries
XML Encryption / Decryption Component
XML Parsing Libraries
XML Signature / Verification Component

Figure 1: General Diagram of the Framework

4.1.1 XML Encryption and Signing Component

In the Encryption and Signing process, there can be a possible need of counter signature, or partial encryption, the framework proposes that it is better to performs 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. XML Security Frameworks gets the key for signature/encryption 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, as shown in figure 2. 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 framework API will have to generate a stencil by itself gathering information by itself. This logic allows application to create stencils without using XML Security Framework functions. Also in some cases stencil should be inserted in the signed or encrypted data (for example, if you want 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 (fig 4)[7].

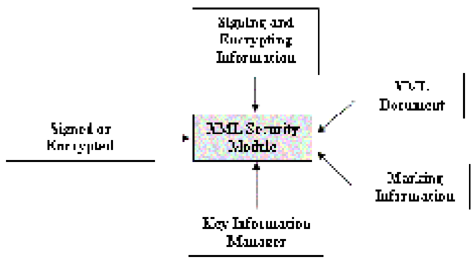

Figure 2: Signing or Encryption module

4.1.2 XML Transformation Component

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. 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. XML Security Framework extends this definition and names "transform" 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 callbacks: "push binary", "push XML", "pop binary" or "pop XML".

In order to simplify transforms development, additional "execute" callback is added. This callback updates internal transform buffers and is used by the "default" XML/binary push and pop callbacks, as shown in figure 3. For example, most of the crypto transforms could be implemented by just implementing one "execute" callback. However, in some cases using push/pop callbacks is more efficient. When necessary, XML Security Framework constructs a transforms chain 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.

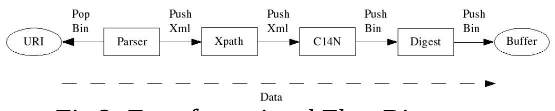

Fig 3: Transformational Flow Diagram

The XML Security Framework transforms engine makes sure that output data type (binary or XML) of previous transform matches the input data type of the next transform by inserting XML

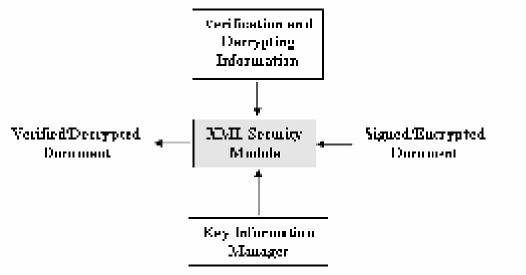
Verification and Decrypting Information
Verified/Decrypted Document
XML Security Module
Signed/Encrypted Document
Key Information Manager

Figure 4: Decryption or Verification Module

parser or default C14N when necessary [8]. Custom transforms could be added by the crypto plug-in or application at any time.

4.1.3 Key Management Component

A key in XML Security Framework is a representation of the <dsig: KeyInfo/> element and consist of several key data objects. The "value" key data usually contains raw key material (or handlers to key material) required to execute particular crypto transform. Other key data objects may contain any additional information about the key. All the key data objects in the key are associated with the same key material. For example, if a DSA key material has both an X509 certificate and a PGP data associated with it then such a key can have a DSA key "value" and two key data objects for X509 certificate and PGP key data, shown in figure 5. XML Security Framework has several "invisible" key data classes. These classes never show up in the keys data list of a key but are used for <dsig:KeyInfo/> children processing (<dsig:KeyName/>, <dsig:EncryptedKey/>, ...). As with transforms, application might add any new key data objects or replace the default ones.

Key Managers

Processing some of the key data objects require additional information which is global across the application (or in the particular area of the application). For example, X509 certificates processing require a common list of trusted certificates to be available. XML Security Framework keeps all the common information for key data processing in a a collection of key data stores called "keys manager":

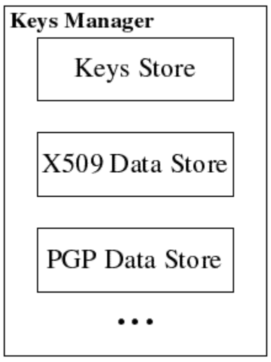

Figure 5: Key Information framework Structure

Keys manager has a special "keys store" which lists the keys known to the application. This "keys store" is used by XML Security Framework to lookup keys by name, type and crypto algorithm (for example, during <dsig: KeyName/> processing). The XML Security Framework provides default "flat list" based implementation of a simple keys store. The application can replace it with any other keys store (for example, based on an SQL database). Keys manager is the only of component in Level 2 of the XML Security Framework which is supposed to be shared by many different components. Usually keys manager is initialized once at the application startup and later is used by XML Security Framework 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.

Inter-Process Communication Component

Application may control XML Security Framework engines behavior using several context objects. New context objects are created for each operation and could not be reused. XML Security Framework also uses context objects to store temporary data and return additional information to the application.

5.0 Summary

This concluded the high/middle level design of the framework, there are a lot of issues will come during the exact implantation of the framework. A portion of the framework is currently being implemented. There is still a lot that can be done on this framework. The features of the framework that are lacking to be discussed in this paper are access control component and digital rights protection component.

The XML Security framework is not the exact solution to the problems of XML based 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 this framework will improve as implementation will be done completely and as maturity level increase it may one day can act as the exact solution.

The planned future work for this paper is to implement the framework up to its current status and then apply various software testing schemes to make the implementation mature, efficient and reliable. Other research oriented areas in relation to this paper can be the up gradation of the framework to satisfy the upcoming security needs like Access Control and Rights Management and also improving the already implemented models.

6.0 References

[1] A. 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] H-ref: "XML Encryption" W3C XML Encryption Working Group, *http://www.w3c.org/Encryption.html*

[8] H-ref: "XML Digital Signature" W3C XML Digital Signature Working Group, *http://www.w3c.org/Signature.html*

[9] H-ref: "XKMS" W3C XKMS Working Group, *http://www.w3.org/2001/XKMS*

[10] H-ref: "XKMS and the Microsoft.Net framework," *http://www.XMLtrustcenter.org/xkms/dotnet/index.htm*