XML ENCRYPTION

BY

Pranay Manek

Email id- pranay_manek2006@yahoo.co.in

Contact - 9686787775

Stream - CPT

Branch - CSE

College - K. L. E. Institute of Technology, Hubli.

&

Mansi Thakkar

Email id- mansithakkar45@yahoo.com

Contact - 9036349575

Stream - CPT

Branch - CSE

College - K. L. E. Institute of Technology, Hubli.

Guided By

Prof. Pradeep Surasura

K. L. E. Institute of Technology, Hubli.

XML ENCRYPTION

Pranay Manek
Computer Science and Engineering
K. L. E. Institute of Technology,
Hubli India.
pranay_manek2006@yahoo.co.in

Mansi Thakkar
Computer Science and Engineering
K. L. E. Institute of Technology,
Hubli India.
mansithakkar45@yahoo.com

Abstract- Internet a most commonly used service on computers today as many services for example online recharge, shopping, internet banking etc. using credit or debit card number and VCC number behind the card, bank account number, passwords etc. are sensitive data so there is need to encrypt the sensitive data. As XML Encryption is a W3C standard for encrypting XML elements. The encryption process involves taking an element from an xml document, encrypting it and it's children, and then replacing the original XML content with the generated encrypted XML in such a way as the document remains well formed. The main goal of XML Encryption is to secure only part of the data, which is sensitive, and provide an efficient and flexible way to transfer data over secure connection. The problem with encrypting the entire document is that all of the document structure gets encrypted. This means that a program must be able to decrypt the entire document before it can make decisions as to how the data should be handled. This is not very efficient. Ideally, only the bits that are truly important need to be secured. The paper submitted by us basically deals with use of XML to encrypt the data and decrypt the data at appropriate receiving user and also sometimes helps to standardize or internationalize the data. XML Encryption is the technology, which concentrates mainly on transferring Data in the most Efficient, Flexible and Secured way across different Platforms and Operating systems.

Keywords: *Asymmetric, Decryption, Encryption, Internet Engineering Task Force, Java Cryptography Encryption, Public and Private Key, Standard Generalized Markup Language, Symmetric, W3C, XML.*

1. INTRODUCTION

1.1 XML

XML (Extensible Markup Language) is a standard for creating markup languages, which describe the structure of data. It is not a fixed set of elements like HTML, but rather, it is like SGML (Standard Generalized Markup Language) in that it is a metalanguage, or a language for describing languages. Extensible Markup Language allows specific markup to be created for specific data. It has the virtues of HTML without any of its limitations. XML is strong in:-Intelligence, Adaptation, Maintenance, Linking, Simplicity, and Portability. XML is intelligent to any level of complexity.

Few of the Properties of XML

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- It should be easy to write programs, which process XML documents.
- XML documents should be human-legible and reasonably clear.

The best interpretation of XML is that it is easy to write a parser for whatever interpretation you want to use, and thus there is a simple way to craft a mechanism for passing information from one program to another.

1.2 Applications of XML in Industry strength

Applications XML is used to Exchange Data

With XML, data can be exchanged between incompatible systems. In the real world, computer systems and databases contain data in incompatible formats. One of the most time-consuming challenges for developers has been to exchange data between such systems over the Internet. Converting the data to XML can greatly reduce this complexity and create data that can be read by many different types of applications.

XML and B2B

With XML, financial information can be exchanged over the Internet.Expect to see a lot about XML and B2B (Business To Business) in the near future. XML is going to be the main language for exchanging financial information between businesses over the Internet.

XML Can be Used to Share Data

Since XML data is stored in plain text format, XML provides a software- and hardware-independent way of sharing data. This makes it much easier to create data that different applications can work with. It also makes it easier to expand or upgrade a system to new operating systems, servers, applications, and new browsers.

XML Can be Used to Store Data

With XML, plain text files can be used to store data. XML can also be used to store data in files or in databases. Applications can be written to store and retrieve information from the store, and generic applications can be used to display the data.

XML is mainly used for Data exchange in applications; EDI (Electronic Data Interchange) and XML go hand-in-hand. When applications are distributed over large networks in big organizations or between different organizations and Data exchange is required, XML can be an efficient and flexible solution for the situation.

Interoperability and data exchange between mainframe systems and other systems running on a variety of hardware and software platforms is essential. XML is a way to generically describe data to facilitate interchange of data between applications. XML output is important for organizations that have the need to exchange data with applications/products running on different platforms or written in different languages.

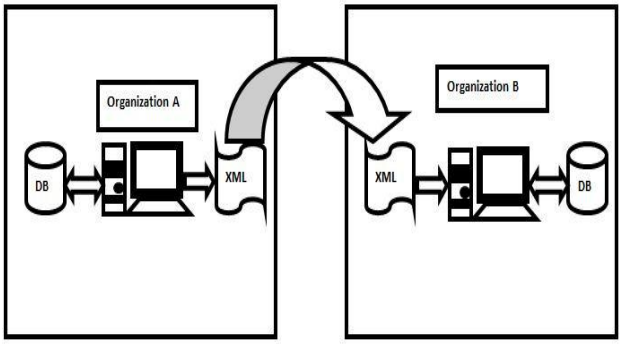

Fig. 2.1 XML on different platforms.

Suppose Organization A works on Linux platform, and the application is built using Java. And Organization B works on the Windows platform and the application is built using Microsofts VB.Net. If application **A** need to exchange some data with application **B**, now this can be achieved only if A and B agree's upon some standard data format. This standard data format can be built using XML. XML DOM and SAX parsers are available both on the Java, as well as on the any MS language.

2. ENCRYPTION

2.1 Introduction

Encryption is the process of converting data from one form into cipher text. The actual process that takes place during this conversion widely varies, but the end result is the same: after conversion to ciphertext, the data is in a form that is not easily readable to prying eyes.

The process of encrypting and decrypting messages has been present since the beginning of primitive communications. Encryption has found many uses over the years, everything from decoder rings in cereal boxes to elaborate methods for governments and corporations to protect their secrets and intellectual property from prying eyes. Encryption helps provide a method to add a degree of security to communications.

Keys

In the quest for a more secure method of protecting information, the introduction of a key adds another level of security. A key is a piece of information that allows only those that hold it to encode and decode a message. Keys come in many different forms such as passwords, numbers generated by an algorithm, digital fingerprints and even electronic devices that work like door keys. It is a series of numbers or symbols that are used to encode a message so that someone in possession of that key or a related key can only read it. A key allows both the sender and the recipient of the message to understand how the message has been encrypted and assures them that nobody else knows how it has been encrypted. It is the key that enables the recipient to properly decode the message. The individual who is sending the message communicates the key to the recipient of the message, allowing them to unlock it. One disadvantage of this system is that an attacker can decrypt the message if the key is intercepted. To protect the key, encryption can be used during communication or the key can be sent in a separate communication.

Symmetric encryption

Uses a single key to encrypt and decrypt the message. After encrypting the data using the key, the same key must be transferred/communicated to the destination for the decryption process.

Asymmetric encryption

Also known as Public-Key encryption uses two different keys - a public key to encrypt the message, and a private key to decrypt it. The public key can only be used to encrypt the message and the private key can only be used to decrypt it. This allows a user to freely distribute his or her public key to people who are likely to want to communicate. To secure information between two users, the sender encrypts the message using the public key of the receiver. The receiver then uses the private key to decrypt the message. Unlike with single or shared keys, in the asymmetric key system only the recipient can decrypt a message; once the sender has encrypted the message

he or she cannot decrypt it. The private key is never distributed, therefore an attacker cannot intercept a key that decrypts the message.

Example of Encryption

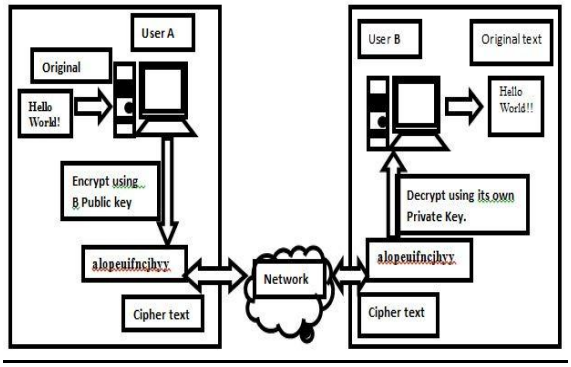

Fig. 2.2 example for encryption.

User A sends Data to User B over the network by encrypting it, using B's Public Key. While the data is on the fly anybody can tap the network line using Network Sniffers and see the data. Since the data is encrypted, it will make no sense. Once the encrypted data reaches its destination, then User B decrypts the received Cipher text using its Private key and retrieves the original text.

3. XML Encryption

3.1 Introduction

XML Encryption is a W3C standard for encrypting XML elements. The encryption process involves taking an element from an xml document, encrypting it and it's children, and then replacing the original XML content with the generated encrypted XML in such a way as the document remains well formed.

There are three approaches to Xml Encryption.

1. Encrypt the xml using **symmetric** encryption: Only one session key is used and it's the same key that encrypts the xml which is used to decrypt it. The key is not stored with the encrypted xml and so the key needs to be loaded during the process and protected when stored.

2. Encrypt the xml using a combination of **asymmetric** and **symmetric** encryption: The dual approach requires a symmetric session key to encrypt the data and an asymmetric key to protect the session key. Both the encrypted session key and the encrypted data are stored together in the xml document. The public asymmetric key is used to encrypt the session key while the private asymmetric key is used to decrypt the key.

3. Encrypt the xml using a **X.509 Certificate:** This approach uses a X.509 certificate as the symmetrical key. X.509 certificates are provided by a third party vendor such as VeriSign.

XML Encryption provides end-to-end security for applications that require secure exchange of structured data. XML itself is the most popular technology for structuring data, and therefore XML - based encryption is the natural way to handle complex requirements for security in data interchange applications.

Currently, Transport Layer Security (TLS) is the de facto standard for secure communication over the Internet. TLS is an end-to-end security protocol that follows the famous Secure Socket Layer (SSL).

Netscape originally designed SSL, and the Internet Engineering Task Force (IETF) later adapted its version 3.0 while they were designing TLS. This is a very secure and reliable protocol that provides end-to-end security sessions between two parties. XML Encryption is not intended to replace or supersede SSL/TLS. Rather, it provides a mechanism for security requirements that are not covered by SSL. The following are two important areas not addressed by SSL:

- Encrypting part of the data being exchanged
- Secure sessions between more than two parties

With XML Encryption, 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. For example, think of a secure chat application containing a number of chat rooms with several people in each room. XML-encrypted files can be exchanged between chatting partners so that data intended for one room will not be visible to other rooms. XML Encryption can handle both XML and non-XML (e.g. binary) data.

3.2 XML Encryption, Syntax and Processing

Anybody who uses XML Encryption has to follow standards set by W3C (World Wide Web Consortium). These standards are Syntax of how the XML document should look after Encryption. The Syntax explain various tags essential to carry out Encryption.

```
<EncryptedData Id? Type? MimeType? Encoding?>
  <EncryptionMethod/>?
  <ds:KeyInfo>
    <EncryptedKey>?
    <AgreementMethod>?
    <ds:KeyName>?
    <ds:RetrievalMethod>?
    <ds:*>?
  </ds:KeyInfo>?
  <CipherData>
    <CipherValue>?
    <CipherReference URI?>?
  </CipherData>
  <EncryptionProperties>?
</EncryptedData>
```

Expressed in shorthand form, the EncryptedData element has the following structure (where "?" denotes zero or one occurrence; "+" denotes one or more occurrences; "*" denotes zero or more occurrences; and the empty element tag means the element must be empty)

The CipherData element envelopes or references the raw encrypted data. If enveloping, the raw encrypted data is the CipherValue element's content; if referencing, the CipherReference element's URI attribute points to the location of the raw encrypted data

Example:

Original XML Document

```
<?xml version='1.0'?>
<PaymentInfo
xmlns='http://example.org/paymentv2'>
<Name>John Smith</Name>
```

<CreditCard Limit='5,000' Currency='USD'>

<Number>4019244502775567</Number>

<Issuer>Example Bank</Issuer>

<Expiration>04/02</Expiration>

</CreditCard>

</PaymentInfo>

Encrypting one of the documents Element, will produce the following result

<?xml version='1.0'?>

<PaymentInfo

xmlns='http://example.org/paymentv2'>

<Name>John Smith</Name>

<EncryptedData

Type='http://www.w3.org/2001/04/xmlenc#Element'

xmlns='http://www.w3.org/2001/04/xmlenc#'>

<CipherData>

<CipherValue>A23B45C56</CipherValue>

</CipherData>

</EncryptedData>

</PaymentInfo>

Encrypting one of the documents Element content, will produce the following result

<?xml version='1.0'?>

<PaymentInfo

xmlns='http://example.org/paymentv2'>

<Name>John Smith</Name>

<CreditCard Limit='5,000' Currency='USD'>

<EncryptedData

xmlns='http://www.w3.org/2001/04/xmlenc#'

Type='http://www.w3.org/2001/04/xmlenc#Content'>

<CipherData>

<CipherValue>A23B45C56</CipherValue>

</CipherData>

</EncryptedData>

</CreditCard>

</PaymentInfo>

Syntax for specifying Key information in the Encrypted XML file.

<EncryptedData

xmlns='http://www.w3.org/2001/04/xmlenc#'

Type='http://www.w3.org/2001/04/xmlenc#Element'/>

<EncryptionMethod

Algorithm='http://www.w3.org/2001/04/xmlenc#tripledes-cbc'/>

<ds:KeyInfo

xmlns:ds='http://www.w3.org/2000/09/xmldsig#>

<ds:KeyName>John Smith</ds:KeyName>

</ds:KeyInfo><CipherData><CipherValue>DEADBEEF</CipherValue></CipherData>

</EncryptedData>

The <EncryptedData> element

The EncryptedData element is the root of the XML encrypted data. The EncryptionMethod element is used to specify the symmetric method used when encrypting the data. It does this by using an Algorithm attribute containing a W3 URL that describes the method used. "http://www.w3.org/2001/04/xmlenc#aes256-cbc" indicates the data was encrypted using AES (Rijndael) with a 256k key size. The Key Info element is borrowed from XML Digital Signatures and is used to store information about the symmetric keys. The KeyInfo element can store information about more than one key. The EncryptedKey element and its child elements contain information about one key stored in a KeyInfo element. The EncryptionMethod element of the KeyInfo contains the asymmetric encryption method used to encrypt the session key. It does this using an

Algorithm attribute set to a W3 URL. For example:

http://www.w3.org/2001/04/xmlenc#rsa-1_5

describes that RSA asymmetric encryption was used to encrypt the session key. The KeyName element is an identifier used to find the key. The CipherData and CipherValue elements that are found as part of the EncryptedKey and EncryptedData elements contain the cipher data. The actual cipher data is stored in the CipherValue element. The EncryptedKey element stores the encrypted key, while in the encrypted data is stored in the CipherValue for the EncryptedData element.

XML Encryption Process

The process of XML encryption can be summarized in five steps:

- Select an element in an XML document.
- Encrypt the element using a symmetric encryption key, known as the session key.
- Encrypt the session key using asymmetric encryption.
- Create an EncryptedData element which will contain the encrypted data and the encrypted session key.
- Replace the original element with the EncryptedData element.

XML Decryption Process

The process of decrypting the XML can be summarized into four steps,

- Select the EncryptedData element in an XML document
- Decrypt the session key using an asymmetric key
- Decrypt the cipher data using the unencrypted symmetric encryption.
- Replace the EncryptedData element with the unencrypted element

3.3 Working of XML Encryption

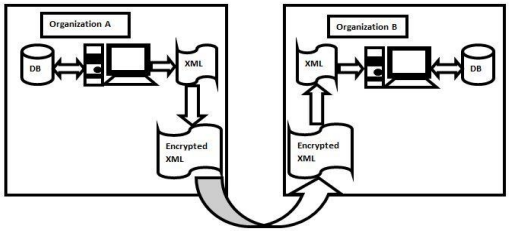

Fig. 3.1 Process of encryption

In XML Encryption solution, the data, which has to be transferred, is first converted to XML, the sensitive data elements are encrypted using any of the encryption techniques and it is then transferred to the destination. At the destination the Encrypted XML file is received, it is then decrypted and used.

4. Case study

Security on MASs with XML Security Specifications:

Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA'06)

0-7695-2641-1/06 $20.00 © 2006 IEEE

In the past, Multi-agent Systems were used in proprietary environments. Nowadays, these systems have been used broadly in open distributed networks, such as e-commerce applications for Internet. An

environment such as the Internet cannot be considered a safe place. Thus, multi-agent systems should have security mechanisms, e.g. confidentiality and integrity. The XML Security Specifications are standards that are based on XML and provide security mechanisms. They include: XML Digital Signature for digital signature; XML Encryption for cryptography; XML Key Management Specifications for Public Key Infrastructure. Agents may use a FIPA standard called RDF, which is a message content standard in XML language. Using this standard, agents can communicate exchanging XML messages, but these messages are not secure. In this article, we propose a secure communication model for agents based on RDF and the XML security Specifications.

Multi-agent Systems (MASs) are types of distributed systems which consist of autonomous entities called agents. The agents are software entities that have enough autonomy and "intelligence" to carry out various tasks with little or no human intervention. When deployed in wide-area networks, such as the Internet, agents can access remote service providers, search for information on the web, and carry out Sale transactions. Agent-mediated electronic commerce is seen as a major application area of agent technology. Unfortunately, an open environment like the Internet cannot be considered safe and reliable. External entities have many methods (attack practices) to interact negatively with an agent. Thus, MASs should have security mechanisms that their agents can use to defend themselves

The prominent security standards and specifications include:

- XML Signature
- XML Encryption
- XML Key Management Specification (XKMS).

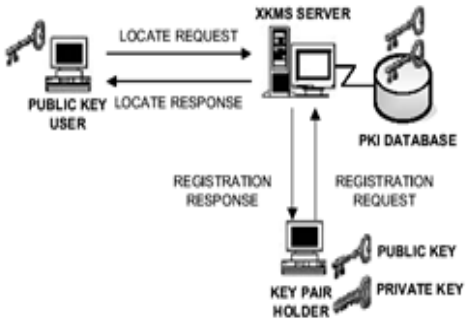

Fig. 4.1 use of Public and Private key

In the proposed model, the agents initiate their activities and generate one key pair. Only one key pair per agent is provided and it consists of a public key and a private key. Immediately, after the agents generate their key pair, they discriminate the public key from the private key. The role of the public key is to be distributed to other agent that participates in the secure communication.

The role of the private key is to be kept in secret by its owner. Since agents need to know receiver's public key to transmit an encrypted message, a XKMS server can be made available to all agents. The role of the XKMS server is to receive various key requests and then it returns the appropriated responses. When an agent generates its key pair, it can register its public key with a

XKMS server to enable other agents to locate and use its registered public key.

To register a key, an agent makes a key registration request and sends it to a XKMS server. The server uses the request and stores a data pair with the key owner and the received public key. Any agent can locate a registered public key from another agent in a XKMS server.

To locate a key, an agent makes a key location request of the agent identification of the desired public key and sends it to a XKMS server. The server searches the PKI database and looks for the public key correspondent to agent identification in the content of the request. If the server had already the public key registered, it responses to the agent with the requested public key information.

The key pair is used to encryption and decryption process of the messages. A message (or a data stream) encrypted by a public key only can be decrypted by using the private key of the same key pair. An agent that needs to transmit secure encrypted message (or a data stream) must know and use the receiver's public key. If the agent does not know the receiver's public key, he can locate the public key. All the agents must register theirs public keys to enable all other agents to locate their public keys. An agent that receives an encrypted message can to decrypt it using its own private key.This model adapted the XML Encryption Specification to provide the encryption and decryption primitives.

5. XML Encryption on different platforms

XML Encryption on Microsoft Platform

The classes needed to perform XML Encryption can be found in three namespaces.

System.Xml – contains XML classes that are needed to contain XML data.

System.Security.Cryptography – contains encryption classes used to generate encryption keys.

System.Security.Cryptography.Xml– contains XML Encryption classes that are used to perform the encryption.

XML Encryption on the Java Platform

The Java Cryptographic Architecture (JCA) Java offers complete support for cryptography. For this purpose, there are several packages inside J2SE, covering all the main features of security architecture such as access controls, signatures, certificates, key pairs, key stores, and message digests.

The primary principle of JCA design is to separate cryptographic concepts from algorithmic implementations, so that different vendors can offer their tools within the JCA framework.

JCA Engine classes

JCA defines a series of Engine classes, where each Engine provides a cryptographic function. For example, there are several different standards of MD (Message Digest) algorithm. All these standards differ in the

implementation, but at the Engine API level they are all the same.

Java Cryptographic Extension (**JCE**)

All independent (third party) vendor implementations of cryptographic algorithms are called Java Cryptographic Extensions (JCEs). Sun Microsystems has also provided an implementation of JCE. Whenever we use JCE, we need to configure it with JCA. For this, we need to do the following:

1. Add the address of the jar file to configure the provided in the CLASSPATH environment variables.
2. Configure the provider in the list of your approved providers by editing the java.security file. This file is located in JavaHome/jre/lib/security folder. The following is the syntax to specify the priority:

security.provider.<n>=<masterClassName>.

Here, n is the priority number (1, 2, 3, etc.). MasterClassName is the name of master class to which the engine classes will call for a specific algorithm implementation. The provider's documentation will specify its master class name. For example, consider the following entries in a java.security file:

- security.provider.1=sun.security.provider.Sun
- security.provider.2=com.sun.rsajca.Provider
- security.provider.3=com.sun.net.ssl.internal.ssl.Provider

These entries mean that the engine class will search for any algorithm implementation in the above mentioned order. It will execute the implementation found first.

6. Conclusion

XML is a very flexible and efficient way to transfer data between applications, Encryption ensures reliable data transfer.

With XML alone we cannot promise transfer of data in an efficient, flexible and secured way. Similarly Encryption alone cannot do the same.

Hence XML Encryption is the technology, which concentrates mainly on transferring Data in the most Efficient, Flexible and Secured way across different Platforms and Operating systems.

References

[1] Applying XML Signature and XML Encryption to Peer-to-Peer Platform Security1-4244-0667-6/07/$25.00 © 2007 IEEE

[2] Proceedings of the 2002 Symposium on Applications and the Internet (SAINT.02w) 0-7695-1450-2/02 © 2002 IEEE

[3] IEEE CCECE/CCGEI, Ottawa, May 2006 1-4244-0038-4 2006

[4] XML Encryption Syntax and Processing W3C (W3C Recommendation 10 December 2002) http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/

[5] Exploring XML Encryption (IBM Developerworks) http://www.ibm.com/developerworks/xml/library/x-encrypt/

[6] Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA'06) 0-7695-2641-1/06 $20.00 © 2006 IEEE