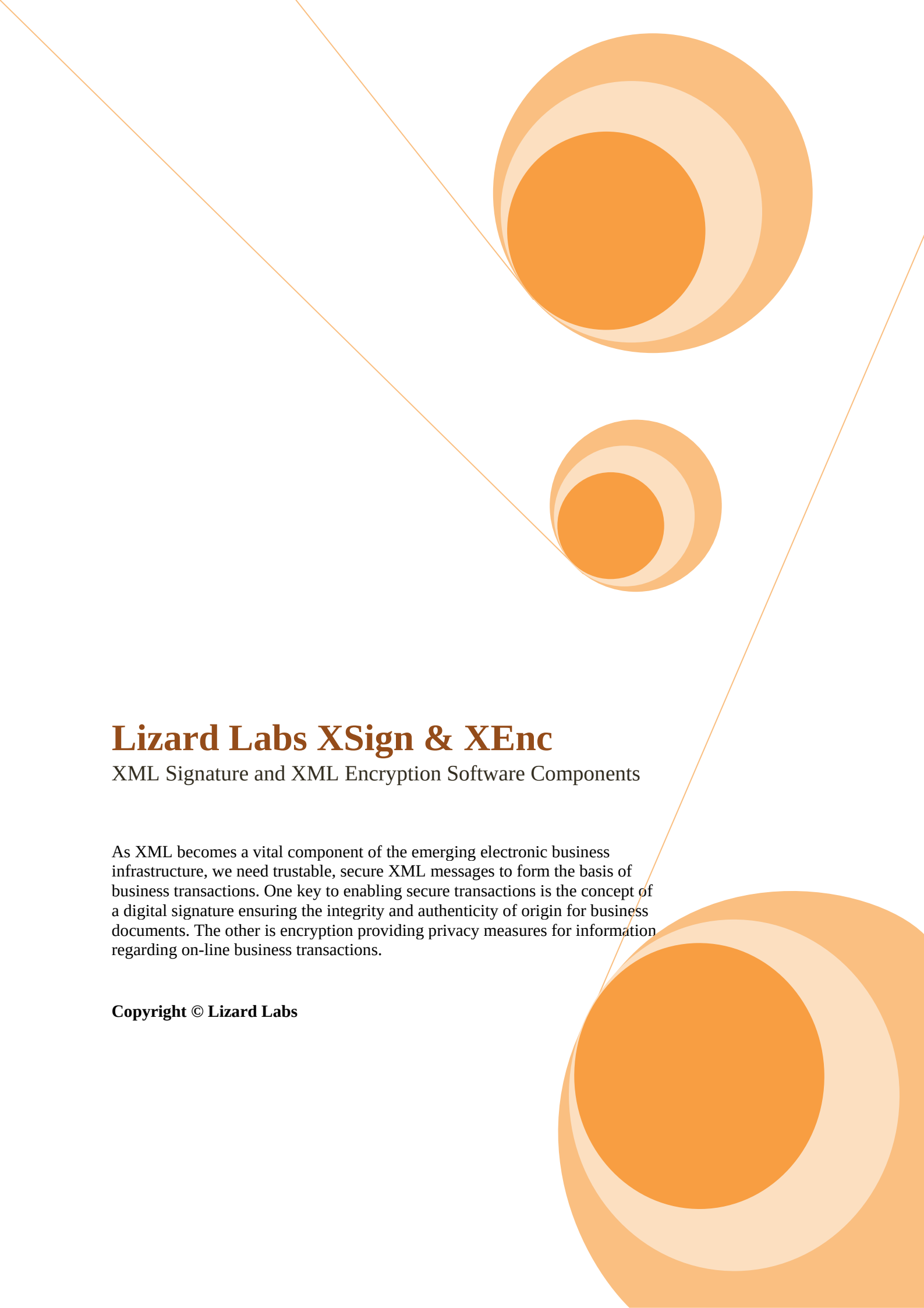


Lizard Labs XSign & XEnc

XML Signature and XML Encryption Software Components

As XML becomes a vital component of the emerging electronic business infrastructure, we need trustable, secure XML messages to form the basis of business transactions. One key to enabling secure transactions is the concept of a digital signature ensuring the integrity and authenticity of origin for business documents. The other is encryption providing privacy measures for information regarding on-line business transactions.

Contents

Notification

“XSign XML Signature Component” and “XEnc XML Encryption Component” (XSign and XEnc in short) are products developed by Lizard Labs. Possession or distribution of these products without a license is illegal. Note that “XSign ActiveX”, “XSign for Firefox”, “XSign ActiveX for Windows Mobile”, “XEnc ActiveX” and “XEnc for Firefox” are all separate products. If you want to use them in your project(s) you'll need a separate license for each one. To view prices and obtain your licenses please visit our site at www.lizard-labs.net or contact us for more information.

About Lizard Labs

Lizard Labs is a European-based research and development team specialized in application and component development, e-business solutions and IT-consulting services. Our products range from professional system utilities to components and libraries for use in software development. We concentrate on the needs of our customers and our software offers unsurpassed quality and usability. Lizard Labs products are compliant to most commonly used security standards defined by W3C, IETF and ETSI. The high quality of our products contributes to long-term satisfaction of our customers and their competitiveness.

Contact Information:

Lizard Labs
Skopska 48
2300 Kocani
Macedonia
E-mail: support@lizard-labs.net
Web: www.lizard-labs.net

About XML Signature and XML Encryption

Security is vital to online business. Technologies designed to meet security requirements have evolved, but the requirements have remained relatively constant. These requirements include Authentication, Authorization, Integrity, Signature, Confidentiality, Privacy and Digital Rights Management. We provide you with top quality components XEnc for encryption and XSign for digital data signing that meet these requirements while implementing XML digital signature and XML encryption software standards recommended by W3C and IETF.

What is XML Signature?

XML enables production and exchange of structured data. The usefulness of such structured data in business critical applications depends upon our ability to assess its origin and authenticity. Digital signatures provide integrity, message authentication, and signer authentication services and are essential to the success of Web Services.

“XML Signature” is a W3C (World Wide Web Consortium) recommendation that defines an XML syntax for digital signatures. Functionally, it has much in common with PKCS#7 but is more extensible and geared towards signing XML documents. It is used by various Web technologies such as SOAP, SAML, and others. It is also known as XML Digital Signature, XMLDsig, XML-DSig, and XML-Sig.

An XML signature used to sign a resource outside its containing XML document is called a detached signature; if it is used to sign some part of its containing document, it is called an enveloped signature; if it contains the signed data within itself it is called an enveloping signature.

Currently XSign implements enveloped signatures.

An XML Signature consists of a Signature element in the http://www.w3.org/2000/09/xmldsig# namespace. The basic structure is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<Signature>
  <SignedInfo>
    <SignatureMethod />
    <CanonicalizationMethod />
    <Reference>
      <Transforms>
        <DigestMethod>
          <DigestValue>
    </Reference>
    <Reference /> etc.
  </SignedInfo>
  <SignatureValue />
  <KeyInfo />
  <Object />
</Signature>
```

- The SignedInfo element contains or references the signed data and specifies what algorithms are used.

- The SignatureMethod and CanonicalizationMethod elements are used by the SignatureValue element and are included in SignedInfo to protect them from tampering. XSign uses RSA SHA1 algorithm (http://www.w3.org/2000/09/xmldsig#rsa-sha1) and Canonical XML Version 1.0 (http://www.w3.org/TR/2001/REC-xml-c14n-20010315)

- One or more Reference elements specify the resource being signed by URI reference; and any transforms to be applied to the resource prior to signing. XSign uses URI="" that identifies the node-set (minus any comment nodes) of the XML resource containing the signature. This means that the root element of XML document is signed.

- DigestMethod specifies the hash algorithm before applying the hash. Only SHA1 is implemented as hash algorithm in XSign.

- DigestValue contains the result of applying the hash algorithm to the transformed resource(s).

- The SignatureValue element contains the Base64 encoded signature result - the signature generated with the parameters specified in the SignatureMethod element - of the SignedInfo element after applying the algorithm specified by the CanonicalizationMethod.

- KeyInfo element optionally allows the signer to provide recipients with the key that validates the signature, usually in the form of one or more X.509 digital certificates. The relying party must identify the key from context if KeyInfo is not present.

- The Object element (optional) contains the signed data if this is an enveloping signature. Enveloping signatures are not supported in XSign.

Digital signatures are created by performing an operation on information such that others can confirm both the identity of the signer, and the fidelity of the information. This capability is important to a growing number of XML protocol, publishing and commerce applications. The XSign component is based on the XML-signature standard of the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF).

For more information about XML digital signatures, see the World Wide Web Consortium (W3C) specification at www.w3.org/TR/xmldsig-core/.

What is XML Encryption?

XML Encryption (also known as XML-Enc) is a W3C standard for encrypting XML elements. The data encryption process involves taking an element from an XML document, encrypting it and its children, and then replacing the original XML content with the generated encrypted XML in such a way as the document remains well formed and can be treated as XML. Although XML Encryption can be used to encrypt any kind of data, it is nonetheless known as "XML Encryption" because an XML element (either an EncryptedData or EncryptedKey element) contains or refers to the cipher text, keying information, and algorithms.

XEnc encryption software provides methods to encrypt the XML element 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. The public asymmetric key is used to encrypt the session key while the private asymmetric key is used to decrypt the key. XEnc covers this approach using a X.509 Certificate as the symmetrical key. X.509 certificates are provided by a third party vendor such as VeriSign.

When you encrypt data using XML Encryption, the result is an EncryptedData element that's included in an XML document. Both the encrypted session key and the encrypted data are stored together in this element. The EncryptedData element contains a series of child elements that describe the algorithms used during the encryption process, as well as containing key information and the cipher data. You can use XML Encryption to encrypt XML documents, XML elements, XML element content, or any other kind of data. Depending on what data you encrypted there are two possibilities:

- If you encrypted an XML element, then the EncryptedData element replaces the element you encrypted.
- If you encrypted the content of an XML element, then the EncryptedData element replaces the element content you encrypted.
- If you encrypted an XML document or arbitrary data, then EncryptedData is the root element of a new XML document or becomes a child element of a document supplied by your application.

XEnc supports first two scenarios. You can encrypt data (element or content) stored in a valid XML document. If you want to encrypt entire XML document you can encrypt its root element.

For more information about XML Encryption Syntax and Processing, see the World Wide Web Consortium (W3C) specification at www.w3.org/TR/xmlenc-core/

Sample XML documents

This is a sample XML document

```
<LIZARDLABSXML:Document
xmlns:LIZARDLABSXML="http://ebank.lizardlabs.com.mk/schemas/LIZARDLABSXML.x
sd">
  <PaymentRq>
    <ObjectGUID>D7748FBA004044A28EDB394176F8ABB0</ObjectGUID>
    <PaymentInstrument>0</PaymentInstrument>
    <ProcessingDate>2007-09-24T00:00:00.0000000+02:00</ProcessingDate>
    <Payer>
      <Name>Dimce Kuzmanov</Name>
      <BankName>Natioanl Bank</BankName>
      <Account>500001004360787</Account>
      <RefNo>123</RefNo>
    </Payer>
    <Reason>money transfer</Reason>
    <Payee>
      <Name>Dimce Kuzmanov</Name>
      <BankName>BANK OF MACEDONIA</BankName>
      <Account>800001004360787</Account>
      <RefNo>123</RefNo>
    </Payee>
    <Amount>10000.00</Amount>
    <PP50Accounts />
    <PP30ReasonCode>410</PP30ReasonCode>
    <SubmissionDate>2007-09-24T12:40:00.0000000+02:00</SubmissionDate>
    <Place>Kocani</Place>
    <PaymentType>I</PaymentType>
  </PaymentRq>
</LIZARDLABSXML:Document>
```

This is the same XML document signed with XSign. The signature is attached to the end of a document. Some element values in these examples are shortened for practical reasons:

```
<LIZARDLABSXML:Document
xmlns:LIZARDLABSXML="http://ebank.lizardlabs.com.mk/schemas/LIZARDLABSXML.x
sd">
  <PaymentRq>
    <ObjectGUID>D7748FBA004044A28EDB394176F8ABB0</ObjectGUID>
    <PaymentInstrument>0</PaymentInstrument>
    <ProcessingDate>2007-09-24T00:00:00.0000000+02:00</ProcessingDate>
    <Payer>
      <Name>Dimce Kuzmanov</Name>
      <BankName>Natioanl Bank</BankName>
      <Account>500001004360787</Account>
      <RefNo>123</RefNo>
    </Payer>
    <Reason>money transfer</Reason>
    <Payee>
      <Name>Dimce Kuzmanov</Name>
      <BankName>BANK OF MACEDONIA</BankName>
      <Account>800001004360787</Account>
      <RefNo>123</RefNo>
    </Payee>
    <Amount>10000.00</Amount>
    <PP50Accounts />
    <PP30ReasonCode>410</PP30ReasonCode>
    <SubmissionDate>2007-09-24T12:40:00.0000000+02:00</SubmissionDate>
    <Place>Kocani</Place>
    <PaymentType>I</PaymentType>
  </PaymentRq>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#"
xmlns:LIZARDLABSXML="http://ebank.lizardlabs.com.mk/schemas/LIZARDLABSXML.x
sd">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315" />
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1" />
      <Reference URI="">
        <Transforms>
          <Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <DigestValue>GugNd7/PQJluZjryrt/6aCbfG0w=</DigestValue>
      </Reference>
    </SignedInfo>

<SignatureValue>MTU3Fb9MWMcisWDkxNrLErKUkYDkwhGk0CT5RCFotlCX2bviPt10Byx6SwO
AY=</SignatureValue>
    <KeyInfo>
      <KeyValue>
        <RSAKeyValue>

<Modulus>ygrvoKB9SUHMvR4tEF9dE1i6bm9Y7vrs/NPOgwa3xV61hk+IybTZ7ZX3OPm6gSVUc=
</Modulus>
          <Exponent>AQAB</Exponent>
        </RSAKeyValue>
      </KeyValue>
      <X509Data>
```

```
<X509Certificate>MIIFfzCCA2egAwIBAgIBAjANBgkqhkiG9w0zNbcVRK+Kj7m4Au8JODn8Gb
zfAWmG/1NS1s7ocQYZErHcDvBqouRlrA==</X509Certificate>
      </X509Data>
    </KeyInfo>
  </Signature>
</LIZARDLABSXML:Document>
```

This is encrypted XML document. In this example PaymentRq element is encrypted:

```
<?xml version="1.0" encoding="utf-8"?>
<LIZARDLABSXML:Document
xmlns:LIZARDLABSXML="http://ebank.lizardlabs.com.mk/schemas/LIZARDLABSXML.x
sd">
  <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
Type="http://www.w3.org/2001/04/xmlenc#Element">
    <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
    <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
        <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
        <xenc:CipherData>

<xenc:CipherValue>IK9wCH8HIEonZoBtRgeEBgSP8de5DaEsKubrBpEvwLFVsSZqw24jy5aA4
WuvKkZdtNViD8o2crctduzvOS9L3VHM8UsNd5RU=</xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedKey>
      <ds:X509Data>
<ds:X509Certificate>MIIFfzCCA2egAwIBAgIBAjANBgkqhkiG9w0BAQUFADByMQswCQYDVQQ
GEwJNSzELMAkGA1UECBMCTUsxFDASBgNVBAoTC0+mJEhrP1+wL1zzPanuuhwuiwnIYUqSrmDv0l
CWfnsrarArSPHmorU+EZuiqfzLH/+TBmnwp+Owc9TORLRCowAyxYuHWhvzXq/DPi04Rx7WKtx/C
9zSIZi+8VO560FH8figzNbcVRK+Kj7m4Au8JODn8GbzfAWmG/1NS1s7ocQYZErHcDvBqouRlrA=
=</ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
    <xenc:CipherData>
<xenc:CipherValue>dFkVzYalOm+InoCt/tjGh/6KvdGmMwqfw1o3bOnueIV9Y0nxvlZztvyw8
GWSCDBfv56sf4Awj9ynKl7tlf0zz3j2cI9gEc0AYy9o52fmfQL8FE5W+qhD8mlbcFAw6GkHQYnd
U3lWphohwmd5kSc1RAPK1AMGt7LpRLNHQfEX TR0H7I6QQ0P1S38k</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
</LIZARDLABSXML:Document>
```

In this example content of the element PaymentRq is encrypted:

```
<LIZARDLABSXML:Document
xmlns:LIZARDLABSXML="http://ebank.lizardlabs.com.mk/schemas/LIZARDLABSXML.x
sd">
  <PaymentRq>
    <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
Type="http://www.w3.org/2001/04/xmlenc#Content">
      <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
          <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
```

```
          <xenc:CipherData>

<xenc:CipherValue>M+A2a/MrHu0MHvcPpTSpObvb80cYhGiVuPKileTxSg=</xenc:CipherV
alue>
          </xenc:CipherData>
        </xenc:EncryptedKey>
        <ds:X509Data>

<ds:X509Certificate>MIIFfzCCA2egAwIBAgIBAjANBgkqhkiG9w0BAQUFADByMQswCQYzNbc
VRK+Kj7m4Au8JODn8GbzfAWmG/1NS1s7ocQYZErHcDvBqouRlrA==</ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
      <xenc:CipherData>

<xenc:CipherValue>OxQJcEZXE15em5C6oJ16k5VrfFLo245z6TVtUjI6I90DgYC072fQ9l147
6+XXzvAo0pIAJm+0NDpH0rTmi8l70LEC6mFPBtZ02c</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </PaymentRq>
</LIZARDLABSXML:Document>
```

Introduction to XSign and XEnc

Both “XSign XML Signature Component” and “XEnc XML Encryption Component” (XSign and XEnc) are data encryption software components that bring strong security to your data and applications. You can use them to sign, encrypt and decrypt XML documents using X509 certificates in just few lines of code. The result XML documents are portable to any environment that supports XML Encryption and XML Signature.

The components provide powerful solution in the following application areas:

Ensure Integrity: XSign component can sign XML documents using user’s private key stored in X.509 certificate. Unauthorized changes in the signed document will be detected by the receiver. If it is used for authentication, it offers a much stronger security and proof of possession than other methods such as one time password (OTP) authentications.

Protect Confidentiality: XEnc component provides strong encryption of your data using Advance Encryption Standard (AES). An XML document can be encrypted using the public key which can only be decrypted by the private key of the same certificate.

The components encapsulate the complexity of XML signature and encryption process into a very simple interface. As we are demonstrating in samples included in the demo version, it takes a very little programming effort to add strong security into your Web and desktop applications.

The most common application areas for these components are web based applications. The digital signature is created by the user in a web browser and the signature is verified on a web server.

Editions

Both components come in two editions:

- As an ActiveX version for use in desktop and Web applications.
- As XPCOM component for use in Firefox

Both of these components are designed to be used on the client side of Web based applications in any scripting environments, such as Java Script or VB Script, in Internet Explorer and Firefox browsers

You can also use ActiveX components to develop desktop applications on any development environments that support ActiveX objects such as C/C++, Visual Basic 6, C#, VB.Net etc.

XSign is also available as an ActiveX component for Windows Mobile and can be used on client side in Pocket IE on mobile devices (Pocket PC and/or Smartphone).

Features

- Signing, encrypting and decrypting XML documents according to W3C XML cryptography standards using X.509 Certificates.

- Available as ActiveX component and can be used with web based applications, desktop applications and scripting environments such as JavaScript and VB script.
- Also available as XPCOM component for use in Mozilla Firefox.
- Compatible with Microsoft.Net, ASP.Net and many other XML encryption, decryption and signature verification libraries.
- Implemented on the Windows platform and uses Microsoft's standard API for cryptographic functions and XML processing.
- Stand alone installation. No need for MS XML, CAPICOM, Java RT or MS.Net.
- Compatible with Internet Explorer and Firefox on all Windows operation systems including Windows XP, Windows Vista and Windows 7 (x86 and x64). XSign is also available for Windows Mobile (Smart Phone and Pocket PC devices).
- Implements the XML digital signature standard (XSign) and XML encryption standard (XEnc) of the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF)
- Full support of Unicode characters.
- Support for user certificate store to find X509 certificates, meaning that all devices including installed certificates on your computer, smart cards, and USB tokens, can be used to sign and encrypt the XML document.
- When creating signatures the signed XML data can easily be made visible for the user for inspection when required
- Easy to use, yet powerful

About the free trial version

The free trial version of the components includes everything the registered version does, except the license rights to distribute applications you create that use the library. That means you get the complete documentation, code examples and sample applications with fully commented source code.

The free trial version is intended to allow you to design code and test applications that use XSign or XEnc for 20 days. The 20-day trial period begins when you first install the library on your development system. Only registered users are permitted to distribute applications that use the library outside their test or development environments.

When you decide that you like the library and will use it in your projects, all you have to do after purchasing a registered copy of the product is to uninstall trials and install registered versions of the components.

Only registered users are licensed to deploy or distribute applications that use the XSign or XEnc components. To place an order visit the Lizard Labs web site to find a link to the secure online ordering system. See the order form for pricing information. There are three licensing models. There is no difference in functionality in any of these models, only in terms of use.

Licensing Options

To accommodate the needs of everyone, from single web developers to worldwide enterprises, Lizard Labs offers three basic licensing models: the Single Developer License, Server License and Unlimited Server License.

You may register your component for one of these three licensing models:

Single Developer License

This allows using Software[1] by one developer in any development environment except web environment (Software cannot be used in web applications with this license). Developer Licenses are non-transferable between developers. ActiveX or XPCOM can be re-distributed as a part of your Application, but the total annual re-distribution of ActiveX should not exceed 200 (two hundred) copies in any year. If your software will do some automate process not directly operated by user or configured with the intention of multiple people accessing it for any kind of service, then the Server License is required for each computer, where your software will be installed

Server License

Allows to use Software in any development environment for developing web or desktop applications. Allows to use Software on a single network server which has unique domain name or IP address, for example "www.my-server.com", "my-local-server" or "192.12.73.247". License will be valid only for internet/intranet applications located on Server, for example "http://www.your-server.com/somepath/app1", "https://www.your-server.com/somepath/app2" or "http://192.12.73.247/somepath/app3". You may purchase an additional Server License if it is needed. You may distribute or transfer, free of royalties, the redistributable files (and/or any

[1] XSign or XEnc software component

developed software) to the extent that they are used separately on one or more computers on the client/workstation side of the network served by the server. You are granted a limited, royalty-free, non-exclusive right to use the software component to design, develop and test the developed software by any number of developers within your organization.

Unlimited Server License

Same as Server License except you do not need to purchase any additional Server Licenses for network servers operating the developed server software in a same organization. This license allows all developers within the purchasing organization and located at the same business address to develop and deploy and freely distribute desktop (non-Internet) and Internet applications using our components. This license is valid to deploy the redistributable files of a component at unlimited number of production servers within the purchasing organization and all organizations that uses developed product.

***

If you are using XSign or XEnc ActiveX to develop a desktop application, you may purchase one single developer license for every developer in your company. But if you have many developers that will develop software products using XSign or XEnc or in the future you are planning a web project, you may consider purchasing Server License. If you are selling web based software to a different organization, they will also have to purchase Server (or Unlimited Server) License separately from your license that you have used to develop the software.

Unlimited server license is suitable mostly for large sites, hosted on many servers (or clustered environment) or for a multiple applications hosted on different servers. With Unlimited License your customers don’t have to license product separately and they can distribute redistributable files, but only if components are used only in your product and they are permitted to develop their own product around these components. Also with Unlimited License you'll have extended development support. Some of the customers that purchased this license also had requirements for some customizations of the component to suits their needs. Prices and purposes of these customizations are negotiated after completion of transaction.

Deployment of redistributable files of ActiveX or XPCOM components to the client machines (end users of your developed products) is free of royalties in all three cases.

We are recommending the Unlimited Server License, since it has most features and you will be on Lizard Labs special customers list, which will give you extended support and some discounts for future updates to our products.

Distributing component to the client

Deployment of ActiveX components

When you write applications that use XSign and/or XEnc ActiveX components you will need to distribute the components along with your application.

You only need to distribute the LizardLabsXEncLiteAX.dll or LizardLabsXSignLiteAX.dll. You may distribute only files located in Redistributables directory of the installation package. You may not distribute any other of the package files.

The DLLs are COM ActiveX controls written entirely with MFC, ATL and Visual C++. The DLL files are fully self-contained, which means that target machines do not need any other files such as CAPICOM, MSXML, MFC40.DLL, MSVCRT40.DLL or ATL.DLL etc. in order for the DLL to register and work on the machine.

The DLL files can be found in the Redistributables directory of the package.

The ActiveX controls support both the single-threaded apartment and the multi-threaded apartment models.

The signed controls are digitally signed with Microsoft Authenticode technology, with Lizard Labs CA as the certificate verification authority. This helps ensure that the components has not been tampered or modified by parties other than Lizard Labs, and is also useful when the component are used on web pages so that browsers do not complain about security issues (Lizard Labs CA has to be installed in trusted root certificates).

Unlimited server license grants you right to sign the unsigned components (DLL, CAB or XPI) with your own code signing certificate published by trusted certificate authority, and distribute such component to your clients.

Typically, desktop applications that use ActiveX controls are distributed with an install or setup program that will take care of registering the controls on the client machine. If you aren't using a setup program, then you will have to manually register the component on the machine.

For web applications there are three ways of redistributing to your client machines: manually, redistributing with a standalone installer and redistributing with a CAB file.

Manually register the ActiveX component

The registration process referred to here is not related to licensing, it refers to the process of making the component (or control) known to the Windows operating system. If you don't use an install program with this capability built-in, you can register your controls manually. From the Run option in the Windows start menu, or from a command prompt, run the following command to register an ActiveX control:

```
regsvr32.exe "path and filename of the ActiveX DLL file"
```

If you get the "LoadLibrary succeeded" message, then the control is now registered on the machine and is ready for use by applications or in your development environment. If you get the failure message, make sure you specified a correct path and filename, and placed it in quotes if it has spaces in it.

Redistributing with a standalone installer

The package includes a Windows Installer installation file for each component, which installs the component on the target machine. This is the easiest way of redistribution. If you are using the components for the Web based applications you can instruct your clients to download and manually install this package from your web site.

Redistributing with a CAB file

The CAB file makes it possible to redistribute the required components through a Web browser (Internet Explorer). You can use this method if you are using them on a webpage. This package includes a CAB (signed and unsigned) files, which can be used to install the component in a Web browser. To do this, you will have to create a page that references the CAB file through the <OBJECT> tag:

```
<object classid='clsid: D39FCCD2-5187-4272-BC9C-3ABDF0D4660F codebase='
XSignClient.cab#Version=1,0,0,0' style="DISPLAY:none;" > The installation of Lizard Labs components failed. Check your security settings.</object>
```

Make sure that the classid and codebase specified matches the component you are using. The clients will have to set the security settings to their browser to low security to allow downloading and installation of the ActiveX components. This will make client computers vulnerable to other malicious sites, so if you are using components from an Internet or intranet applications, we recommend using a standalone installer by providing a link to download MSI file. The other option is adding the site to trusted security zone in IE and/or using a signed CAB file for distribution. Depending on your product license you can also sign unsigned CAB file with your code signing certificate provided from trusted Certificate Authority.

Internet Explorer client configuration

To use XSign or XEnc in your web based applications, your clients will have to set proper security configuration of their web browsers (Internet Explorer). They will have to allow execution of unsigned ActiveX software components. But if they do that for all Internet sites, their computers will be very vulnerable to malicious code from some other sites and will be easy targets for spammers, viruses and spy ware. So we recommend suggesting that your clients add your web site to a “trusted security zone” and to set all trusted sites to “low security level” this will allow execution of unsigned ActiveX components only for trusted sites, including yours. In order to do this, your clients will have to do following steps:

1. From the Internet Explorer menu select “Tools” and then “Internet Options” menu command
2. In the “Internet Options” dialog box, select “Security” tab
3. In the web content zone window, select “Trusted sites”
4. Click on “Sites...” button and enter the address of your web site and click “Add”, then “Ok”
5. In the security level window, click “Default level” button and with the slider, select low security level

6. Click “Ok” and close internet options dialog.

Deployment of XPI package for Firefox

You have to use this method for clients that are using Firefox to access your web application. Installation package includes signed and unsigned XPI files for each component which can be used to install the component in your clients Firefox browser.

To enable installation and usage of signed XPCOM Lizard Labs CA Certificate has to be installed in Firefox Certificate store (trusted certificates). Your clients can install unsigned version of the component without certificates but security warnings will pop up while using the component. You can sign the unsigned XPI with your own certificate published from a trusted CA.

To enable use of unsigned XPCOM from the web application your clients have to enable codebase_principal_support in their browser preferences. To do so they have to follow these steps:

1. In Firefox location bar type about:config.
2. The new window comes up and type "signed.applets.codebase_principal_support" in the filter bar.
3. The property named: "signed.applets.codebase_principal_support" will be displayed.
4. Select the property and then "right click" on it, select toggle to convert (the Boolean) value to "true". This option adds support for interactive security configuration for signed applets and JavaScript in your browser.

To install the plug-in for the Firefox drag and drop XPI file to the Firefox browser window or open it with the Firefox.

For more information about installing XSign and XEnc on Firefox please refer to the samples for Firefox provided with demo version of the product. Note that ActiveX and XPCOM versions of the component are different products with similar functionality.

Deployment of ActiveX for Windows Mobile

For security reasons Pocket IE does not support automatic downloads and installation of ActiveX cab files. To install the component you (and your customers) will have to copy CAB file (from Windows Mobile directory) on mobile device and click on it to begin installation process.

You may test installation success following these steps:

After installing the component, copy these files on your device:

```
\TestWebApp\TestHTML.htm

\\TestCertificates\LizardLabsTest.pfx
```

If you have Windows Mobile 6.0 (or later), just click on LizardLabsTest.pfx to install certificate in certificate store.

If you have Windows Mobile 5.0 you will have to use external tools to install personal certificate like pfximport (http://www.jacco2.dds.nl/networking/pfximprt.html).

After successfully installation of certificate open TestHTML.htm in Pocket Internet Explorer and click on test button. A dialog box containing the signed XML should appear on the screen.

XSign Reference

The “XSign XML Digital Signature Control” is an object that performs signing task on the well formed XML documents according to W3C XML Signature standard recommendations. This section contains descriptions of methods and properties implemented in XSign Class.

Class Info

Description	Value
ActiveX Class ID	D39FCCD2-5187-4272-BC9C-3ABDF0D4660F
ActiveX Class Name	LizardLabs.XSignLiteCtrl.1
ActiveX Component Name	Lizard Labs XSignAX ActiveX Control Module
XPCOM Class Name	@lizardl.com/xsignlite;1

Properties

Property Name	Type	Description
IsCertificateSelected	VARIANT_BOOL	Read only property. True if signing certificate is selected from certificate store. False otherwise.
Signature	BSTR	Read only property that holds only a value of XML digital signature element. If you want to verify the signature you will have to recreate the original XML document and append the signature
SignatureWithoutKeyInfo	BSTR	Read only property that also holds the value of XML digital signature but without the KeyInfo element. Some security experts are saying that distributing a key info with a signature
SignedXml	BSTR	Read only property that holds the signed XML document after finishing the singing process with method Sing()
SigningAlgorithm	BSTR	Only available on ActiveX component. Get/set signature algorithm. Supported values: “sha1” (default), “sha256”, “sha384” and “sha512”

Methods

Method Name	Return Type	Description
AboutBox()	VOID	Opens a dialog box with information about XSign

Method Name	Return Type	Description
GetCertificatesList()	BSTR	The method returns a string with list of certificates from user's personal certificate store. Every certificate is separated with LF (ASCII value 10, or '\n') and every field of each certificate is separated by TAB (ASCII value 9, or '\t'). The fields included are: IssuedTo, IssuedBy and Thumbprint/Hash. This method is not available in XSign for Firefox and XSign for Windows Mobile.
SelectCertificate()	VARIANT_BOOL	Choosing certificate from personal certificate store. A dialog box is shown with all installed certificates. A private key is required in order to digitally sign XML document. Returns True if certificate is selected, False if it is not.
SelectCertificateByThumbprint (BSTR Thumbprint)	VARIANT_BOOL	Method for choosing signing certificate from personal certificate store by using unique certificate hash (thumbprint). Thumbprint is a part of certificate properties. Returns True if certificate is selected, False if it is not.
Sign(BSTR XmlString)	VARIANT_BOOL	The method signs an XML document using the certificates private key. The method creates an enveloped signature and inserts the signature as the last child of the root element. Notice that a parameter 'XmlString' is a string and not an MS Xml Dom object. Returns True on success or False if error occurred.
SignURI (BSTR XmlString, BSTR URI)	VARIANT_BOOL	XSign can also sign a specific element within an XML document using an enveloped signature. Use the SignUri method to specify which element of the current XML document to sign with parameter URI. The name of XML element is identified by an attribute name ID. The URI string should take the following form #IDName where IDname is the name of a valid attribute name ID. Example use is SignURI("<tst id='abc'>xyz</tst>","#abc").This method is not available in XSign for Firefox and XSign for Windows Mobile.

XEnc Reference

The “XEnc XML Encryption and Decryption Control” is an object that performs encryption and decryption tasks on the valid XML document. This section contains descriptions of methods and properties of XEnc Object.

Class Info

Description	Value
ActiveX Class ID	69AC1298-610C-4260-AB21-760698EA8513
ActiveX Class Name	LizardLabs. XEncLiteCtrl.1
ActiveX Component Name	Lizard Labs XEncAX ActiveX Control Module
XPCOM Class Name	@lizardl.com/ xenclite;1

Properties

Property Name	Type	Description
DecryptedXml	BSTR	Read only property that holds the decrypted XML document after finishing the decryption process with method Decrypt()
EncryptContent	VARIANT_BOOL	Sets element encryption method. If FALSE, entire document element is encrypted, if TRUE, only element content is encrypted. Default value is FALSE.
EncryptedXml	BSTR	Read only property that holds the encrypted XML document after finishing the encryption process with method Encrypt()
IsCertificateSelected	VARIANT_BOOL	Read only property. TRUE if encryption/decryption certificate is selected from certificate store. FALSE otherwise.
SystemSotreName	BSTR	Determines from which certificate store to select certificate for encryption or decryption. Default value is “My” and represents personal store. Other possible values are: “Root”, “CA”, “OtherPeople”, etc…
UseLocalMachineStore	VARIANT_BOOL	If FALSE users certificate stores are used to select required certificate. If TRUE local machine stores are used. Default value is FALSE.

Methods

Method Name	Return Type	Description
AboutBox()	VOID	Opens a dialog box with information about XSign

Method Name	Return Type	Description
SelectCertificate()	VARIANT_BOOL	Choosing certificate from certificate store selected with properties SystemSotreName and UseLocalMachineStore. A dialog box is displayed on screen with all installed certificates. A private key is required in order to decrypt an encrypted XML document. Public key is enough for encryption. Returns True if certificate is selected, False if it is not.
SelectCertificateByThumbprint (BSTR Thumbprint)	VARIANT_BOOL	Method for choosing certificate from certificate store by using unique certificate hash (thumbprint). Thumbprint is a part of certificate properties. Returns True if certificate is selected, False if it is not.
Decrypt(BSTR EncryptedXml)	VARIANT_BOOL	The method decrypts data in an XML document. It will try to process all EncryptedData elements in the document and replace them with the decrypted contents. Notice that a parameter 'EncryptedXml' is a string and not an MS Xml Dom object. The method returns True if the operation succeeded and stores result XML document in DecryptedXml property.
Encrypt(BSTR XmlDoc, BSTR Element)	VARIANT_BOOL	The method encrypts a specific element of the document using XML Encryption. If parameter Element is empty string, a whole document is encrypted. The method returns True if the operation succeeded and stores result XML document in EncryptedXml property. It returns False if something is wrong.

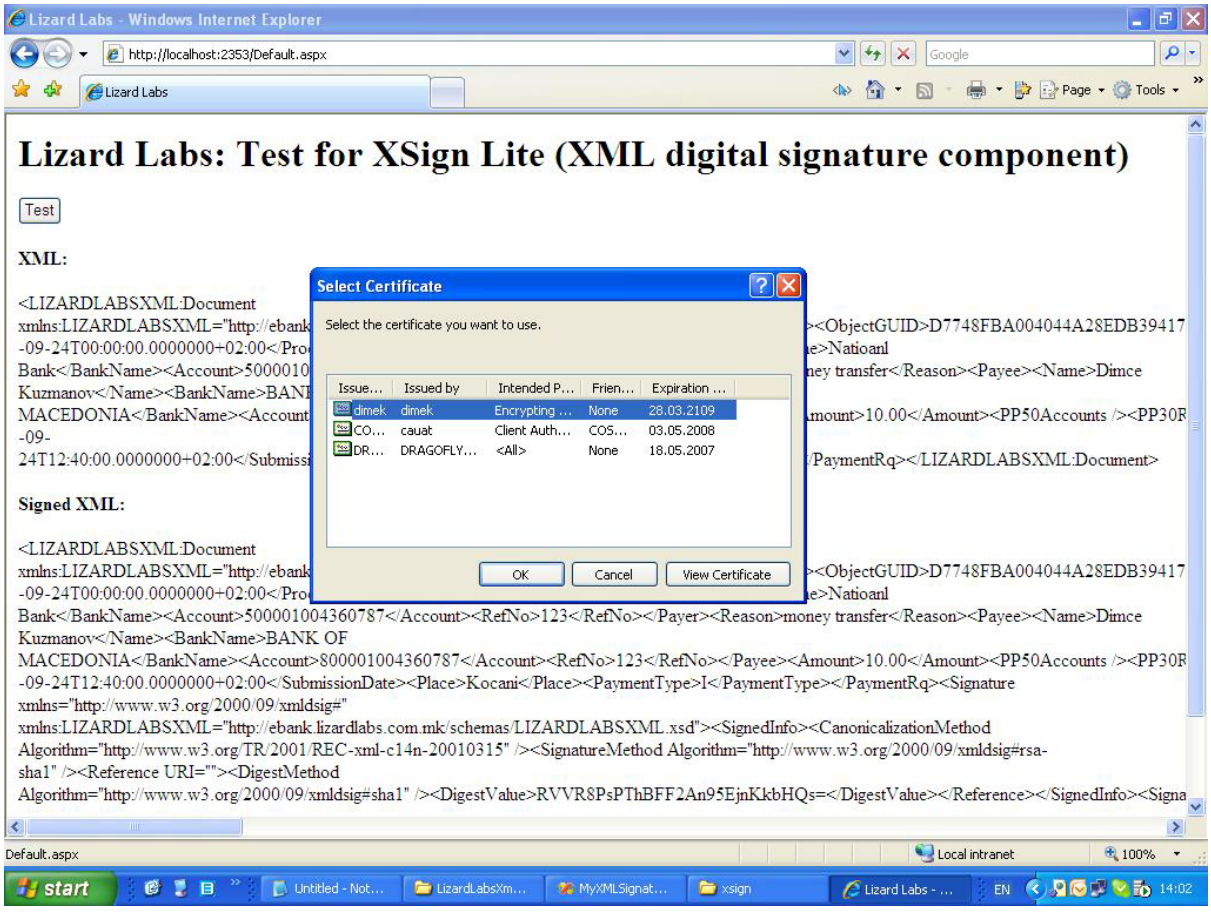
Examples

The following sample code shows how to use XSign and XEnc ActiveX in HTML using JavaScript.

```
<object id="oXsignObject" classid="clsid:D39FCCD2-5187-4272-BC9C-
3ABDF0D4660F" codebase="LizardLabsXSignLite.cab" style="display: none;" />

<object id="oXencObject" classid="clsid:69AC1298-610C-4260-AB21-
760698EA8513" codebase="LizardLabsXEncLite.cab" style="display: none/>

<script language="javascript">
    function TestSign() {
        try {
            oXsign.SelectCertificate();

            if (!oXsign.IsCertificateSelected) {
                alert("Please select valid certificate.");
                return false;
            }
            else {
                if (oXsign.Sign("<test id='abc'>Test xml</test>")) {
                    alert(oXsign.SignedXml);
                    alert(oXsign.Signature);
                    alert(oXsign.SignatureWithoutKeyInfo);
                    return true;
                }
                else alert("Error");
            }
        }
        catch (e) {
            alert("JavaScript Error: " + e.message);
            return false;
        }
        return false;
    }

    function TestEncrypt() {
        try {
            oXenc.UseLocalMachineStore = false; // Use users store
            oXenc.SystemSotreName = "My"; // Use personal store (default).
            oXenc.EncryptContent = false; // Encrypt XML element

            oXenc.SelectCertificate();

            if (oXenc.IsCertificateSelected) {
                if (oXenc.Encrypt("<test>Test xml</test>", ""))
                {
                    alert(oXenc.EncryptedXml);
                    // Decrypt the document
                    // this sample uses the private key for
                    // the same certificate that you have selected
                    // for encryption)
                    if (oXenc.Decrypt(oXenc.EncryptedXml)) {
                        alert(oXenc.DecryptedXml);
```

```
                    }
                    else alert("Error decrypting XML");

                    return true;
                }
                else alert("Error encrypting XML");
            }
            alert("Please select certificate!");
        }
        catch (e) {
            alert("JavaScript Error: " + e.message);
        }
        return false;
    }
</script>
```

Sample screenshot



Legal information

Exporting of software that uses strong encryption

Some countries restrict the exporting of software that uses strong encryption, while other countries freely allow you to export software that uses encryption as long as it meets some conditions such as being widely available for purchase over-the-counter. If you will be exporting software that uses the XSign or XEnc component, we strongly advise you to find out what your country's laws allow or restrict.

Many countries participate in the Wassenaar Agreement. This is an international agreement which includes restrictions on exporting cryptography goods. Their web site is a good place to get started, and will probably point you to your country's web site's encryption laws pages. The URL for their web site is http://www.wassenaar.org.

Official Copyright & Disclaimer

Glossary

	Description
AES	Advanced Encryption Standard. This standard, developed conjointly by the U.S. National Institute of Standards and Technology (NIST) and the international cryptographic community, specifies an encryption algorithm capable of protecting information well into the 21st century. It replaces the outdated Data Encryption Standard (DES).
Certificate	A certificate is an attestation that a particular Public/Private Key belongs to a particular person, company or entity. They are meant to help discourage someone from using a counterfeit key in order to impersonate someone else. A certificate commonly contains the name of a person, company or entity as well as the public key associated with them and information about the organization that issued the certificate. The whole certificate itself is then digitally signed by the certificate issuer to prove that it is genuine and has not been altered by anyone else. Example of popular certificate issuer is VeriSign.
Private Key	One of the two keys of a public-key encryption key pair, the Private Key is the key that is typically held only by one party, the party that created the key pair.
Public Key	One of the two keys of a public-key encryption key pair, the Public Key is the key that is typically distributed at large, and can be held by many parties.
Public-key encryption	Public-key encryption is an encryption technique that involves a pair of keys instead of only one key like with symmetric encryption. The key pair consists of a Public Key and a Private Key that are created to work together. Anything encrypted with one key can only be decrypted by the other key. The Public Key is meant to be distributed at large and the Private Key is meant to be held only by a single party. The Public Key, which can be held by many parties, can be used to encrypt a message to be sent to the holder of the Private Key. Only the holder of the Private Key can decrypt the message. The reverse process (using the Private Key to encrypt data) is useful for digitally signing documents, so that, for example, the holder of the Private Key can use it to encrypt data which only holders of the Public Key can decrypt. The holders of the Public Key can then be sure that the author of the encrypted data can only be the holder of the Private Key,

World Wide Web Consortium (W3C)	The World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web that oversee the development, formulation, maintenance and issuance of common standards for the World Wide Web including the standards governing HTML, XML and CSS to "promote its evolution and ensure its interoperability."
X.509	In cryptography, X.509 is an ITU-T standard for a public key infrastructure (PKI) for single sign-on (SSO) and Privilege Management Infrastructure (PMI). X.509 specifies, amongst other things, standard formats for public key certificates, certificate revocation lists, attribute certificates, and a certification path validation algorithm.