Create a Certificate for Signing a XML Document

After installing OpenSSL

* Copy Path { @C:/PATH/BIN }
* Rt Click PC –Eigenschaften -> erweiterte Systemeinstellungen -> change user -> Umgebungsvariablen -> Path -> Bearbeiten -> Neu -> Paste Path
* CMD Prompt -> openssl then cmd+c
* CMD: cd DESKTOP/Folder
* CMD: openssl req -x509 -days 365 -newkey rsa:2048 -keyout ClientName-key.pem -out ClientName-cert.pem
* Enter Pass Phrase, State, Country, Org and Email
* CMD: openssl pkcs12 -export -in ClientName-cert.pem -inkey ClientName-key.pem -out ClientName-cert.pfx
* Enter password
* Create public cert – CMD: openssl pkcs12 -in ClientName-cert.pfx -clcerts -nokeys -out ClientName-cert-public.pem

Create DesktopApp:

* CreatesignedDocProgram in C#:
* using System;
* using System.Collections.Generic;
* using System.IO;
* using System.Linq;
* using System.Net;
* using System.Security.Cryptography.X509Certificates;
* using System.Security.Cryptography.Xml;
* using System.Text;
* using System.Threading.Tasks;
* using System.Xml;
* namespace CreateSignedDoc
* {
* class Program
* {
* static void Main(string[] args)
* {
* XmlDocument doc = new XmlDocument();
* using (WebClient client = new WebClient())
* {
* byte[] xmlBytes = client.DownloadData("https://www.wibit.net/labWebService/rest/getCoursesForCurriculum/1");
* doc.LoadXml(Encoding.UTF8.GetString(xmlBytes));
* }
* string pfxPath = @"C:\PATH\test-cert.pfx";
* X509Certificate2 cert = new X509Certificate2(File.ReadAllBytes(pfxPath), "test");
* SignXmlDocumentWithCertificate(doc, cert);

* File.WriteAllText(@"C:\PAth\signedXmlDoc.xml", doc.OuterXml);
* Console.WriteLine(doc.OuterXml);
* Console.ReadLine();
* }

* public static void SignXmlDocumentWithCertificate(XmlDocument doc, X509Certificate2 cert)
* {
* var signedXml = new SignedXml(doc);
* signedXml.SigningKey = cert.PrivateKey;
* var reference = new Reference();
* reference.Uri = "";
* reference.AddTransform(new XmlDsigEnvelopedSignatureTransform());
* signedXml.AddReference(reference);
* var keyInfo = new KeyInfo();
* keyInfo.AddClause(new KeyInfoX509Data(cert));
* signedXml.KeyInfo = keyInfo;
* signedXml.ComputeSignature();
* XmlElement xmlSig = signedXml.GetXml();
* doc.DocumentElement.AppendChild(doc.ImportNode(xmlSig, true));
* }
* }
* }

Create ValidateDoc

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Security.Cryptography.Xml;

using System.Text;

using System.Threading.Tasks;

using System.Xml;

namespace ValidateSignedDoc

{

    class Program

    {

        static void Main(string[] args)

        {

            string xmlString = File.ReadAllText(@"C:\PATH\signedXmlDoc.xml");

            XmlDocument doc = new XmlDocument();

            doc.LoadXml(xmlString);

            X509Certificate2 pubCert = new X509Certificate2(@"C:\PATH\test-cert-public.pem");

            Console.WriteLine(ValidateDocument(doc, pubCert));

            Console.ReadLine();

        }

        public static bool ValidateDocument(XmlDocument doc, X509Certificate2 cert)

        {

            try

            {

                SignedXml signedXml = new SignedXml(doc);

                XmlNode signatureNode = doc.GetElementsByTagName("Signature")[0];

                signedXml.LoadXml((XmlElement)signatureNode);

                return signedXml.CheckSignature(cert, true);

            }

            catch

            {

                return false;

            }

        }

    }

}