XML Digital Signature and its Role in Information System Security

Sandro Gerić, Tomislav Vidačić
University of Zagreb
Faculty of Organization and Informatics
Pavlinska 2, Varaždin, Croatia
Phone: +385 (0)42 390 857 Fax: +385 (0)42 213 413
e-mail: sandro.geric@foi.hr, tomislav.vidacic@gmail.com

Abstract - XML signature is form of digital signature designed for use in XML transactions. The XML Digital Signature standard defines a schema that is used for storing the result of a digital signature operation applied to (in most cases) XML data. Like non-XML digital signatures, XML signatures add authentication, data integrity, and support for non-repudiation to the data that is object of XML digital signing process. However, unlike non-XML digital signature standards, XML signature has been designed to both account for and take advantage of the Internet and XML.
A fundamental feature of XML Signature is the ability to sign only specific portions of the XML content rather than the complete document. This is relevant when a single XML document may have a long history in which the different components are authored at different times by different parties, each signing only those elements relevant to it. This flexibility will also be critical in situations where it is important to ensure the integrity of certain portions of an XML document, while leaving open the possibility for other portions of the document to change. Since data security – in form of data verification and authorization - represents an important part of information system security paradigm this article is addressing the questions and possibilities of XMLDigSig usage in everyday information system security procedures.

I. INTRODUCTION

The trends in ICT development and the way the ICT supports business processes are clearly focused on use of web based technologies and web services in the last decade. Web services, as a programming technology, enable companies to use and integrate different application modules (e.g. developed by them or third parties) into one unique information system infrastructure that can be present in form of *real*, but also *virtual*, or *on-demand* information system. In case of web services based information system infrastructure it is important to achieve a desired and demanded level of security, just like in any other form of information system infrastructure. Therefore this article addresses the evolution of web services based architectures and it's focused on deployment and implementation of secure web services.

Just like in other forms of information systems architectures, in web services based information systems the question of reliability and security can be addresses in many different ways – from organizational security measures, technical, program, legislative, etc. But one form of security measures that characterizes web services based information systems is the use of different forms of digital signatures to ensure the integrity of content and to authenticate the author (person, company, application component) of specific content (data, information, service component, application component). One form of digital signature specifically design for the use in web content is XML digital signature, and in this article we will try to determine if the security level of web services messaging mechanisms can be improved through the introduction of XML digital signatures. Namely, the web service's technology and the way web services are "working" is based on exchange of XML messages defined by specific WSDL format unique to each web service.

II. WEB SERVICES AS AN FORM OF INFORMATION SYSTEM INFRASTRUCTURE

The history of information system's architectures development was, and still is, motivated with desire for higher program code re-usability. In that manner different types of information systems architectures were developed with time, and each evolutional step used the current and state-of-the-art technologies. The important step in this process was emergence of Object Oriented paradigm, which introduced concepts like inheritance, polymorphism and data abstraction. Those concepts introduced by Object Oriented (OO) design provided developers with new level of code re-usability. But, despite the positive trends in application development affected in OO design there are some limitations. One of the most important limitation is the fact that the classes used in program code have to be included in it, in other words they have to be accessible in the application framework. To achieve that it is necessary to redesign the application structure, to use and store Application Programming Interface in the class path or to correctly reference compiled DLL (2) To achieve this all parts and program components of an application have to be (physically) stored on one computer, and in the world of business-to-business integration processes and cross-organizational information systems this is serious problem.
The next evolutional step in information system's architectures development was a set of architectures that are based on OO concepts, but more focused on distribution of program components and their interoperability. The architectures like component based architecture (CBA), distributed computing emerged to

response of OO limitations. Component-based development allows developers to create more complex, high quality systems, because it provides better means of managing complexities and dependencies within an application. A software component is defined as a unit of composition with contractually specified interfaces and explicit context dependencies. It can be deployed independently and is subject to composition by third parties It is a group of objects with has a specified interface, working together to provide an application function (9). Similarly to this the idea of distributed computing was to set up a communication between two distributed programs directly on the basic physical network protocol and to use distributed information system resources (8). As the next evolutionary step, in this process a communication middleware framework enables to access a remote application without knowledge of technical details such as operating systems, lower-level information of the network protocol, and the physical network address. As distributed computing technologies evolve, it becomes increasingly necessary to provide multiple network implementations to satisfy various quality-of-service requirements. These requirements may include timeliness of message delivery, performance, throughput, reliability, security and other nonfunctional requirements. (4)

The end of nineties and the evolution of network based resources and web services defined a new step in information system architecture evolution that is characterized with use of network based resources and technologies. The programming technology that lays behind architectures like Service-oriented architecture (SOA), Cloud computing, Software-as-a-Service architecture is web service technology. Web services can be implemented using various programming languages and are therefore heterogeneous from point of their implementation. In order to work and to connect web services implemented in different programming languages each web service (as a program component) has defined an unique and specific web service interface based on web service description language (WSDL). WSDL interface describes all parameters that are necessary for use of specific web service, like XML message format, procedure/ method/ function calls, ports, etc. Since web service exists in network environment a XML based messaging system is used for their communication and control. Therefore, even IBM defines web services as programs that accept requests in XML format from other systems across the Internet or Intranet via lightweight and vendor-neutral communications protocols (IBM developersWorks, 2003a).

The implementation of web service usually consists of following processes. Firstly, the web service that requires access to the another program has to send a method with its arguments (defined by WSDL interface) through a Remote Procedure Call (RPC). This is done using Simple Object Access Protocol (SOAP) (Champion, Ferris, Newcomer and Orchard, 2002). that is developed and used specifically for communication between different web services. Each RPC call is stored in a SOAP message that is constructed using XML specification and structure defined by web service WSDL. That way XML document structure contains all information necessary to start an activity in web service. After the remote application has received the SOAP message it will begin to process the content of the message (it will start to perform a specific operation or activity defined by message) and after the operation is finalized it will send the response back to the caller in XML format (2)

III. SECURITY ISSUES IN MODERN INFORMATION SYSTEM'S ARCHITECTURES

Information system's architectures based on use of web services are widely spread today. They offer many advantages compared to "traditional" form of IS architectures, but also have some concerns, specially from information system security point of view. Modern IS architectures usually consist from more than one services (e.g. web services) that are produced and developed by different organizations and individuals, that are developed under different platforms, and operate in different conditions and under different security management systems. All those components are connected into one information system that has to provide its users with the same level of security as classical IS. And that leads to the mayor security challenges that can be grouped in four main categories (5):

- *Data security and data control,*
- *Development and use,*
- *Authentication,*
- *Requirement for "off-the-shelf" components.*

The focus of this article is on first category – data security and data control. One of the oldest security mechanisms that is used for securing data in context of network based resources, and that has a great popularity in securing electronic transactions is Secure Socket Layer (SSL). SSL is used for securing communication path between two points (user side and server side), and it provides a sufficient level of security during the transport but it leaves data unprotected on origin (user) or destination (server) side.

Although some authors (7) suggest that it is not likely that data would be tempered during transmission; moreover identity management and non-repudiation issues have been looked over in the deployed version of SSL. This has been a concerning issue for companies wanting to deploy web services solutions to perform high value business transactions. (1, 7, 9)

This resulted with a development of a solution that will secure the data all over their transportation path, and not only during the process of transport, and that will be optimized for securing communication between web services, in other words that will be optimized for securing XML based transmission. The result was a new form of digital signature for XML documents developed by W3C and IETS that provides authentication, data integrity, and support for non-repudiation of the signed. It is based on similar algorithms and mechanisms like "standard" forms of digital signatures, but it has a unique characteristic that can be used for signing a specific element of XML

structure, rather than complete document. Just like a "standard" form of digital signature it defines a verification algorithm that indicates the originator of the XML document and thereby revealing the sender's identity. (1)

IV. XML SIGNATURES

XML Digital Signature (XMLDigSig) is a form of digital signature that is optimized for signing of XML data. What differ the XML digital signature mechanism from "standard" digital signatures is possibility of *partial signature* which allows that XML digital signature is used only on specific tags in XML structure. Beside partial signature, XML digital signature also has the possibility of *multiple signatures* that enables signing of multiple tags in XML structure.

The XMLDigSig was developed to solve specific security issues concerning data security and control in electronic transactions, and later in messaging mechanisms used in web services. Primarily, it is used to solve problems like falsification, spoofing, and repudiation.

Comparing with the "standard" digital signatures, the result of XMLDigSig is stored in **<*Signature*>** element expression, and the result of "standard" digital signature is represented by a string of code calculated based on input data. XMLDigSig are suitable for use in the network environment. Standard digital signature mechanism is relaying on Certificate Authority, as a third party used for verifying digital signature certificates and signature itself. In order to do this an extensive network communication had to be conducted what in some cases affected the efficiency of signature confirmation process. The XMLDigSig is using URI modeling that is incorporated into network resources, so the entire process of XMLDigSig verification is more networks efficient.

The result of XMLDigSig is stored in form of XML syntax that uses **<*Signature*>** element. Beside the signature value, this syntax stores all other information relevant to XMLDigSig verification. The syntax is defined by W3C organization, and it's available through web link: http://www.w3.org/TR/xmldsig-core/.

The main element of XMLDigSig specification is **<*Signature*>** element. Its child elements are used to store all information regarding the signature and its verification. **<*SignedInfo*>** element defines references of the algorithm used for the XML signature creation and points out the targeted XML data. It is also used to store digest value and other information. **<*SignatureValue*>** element is used for storing the signature value, and a **<*KeyInfo*>** element contains information about XMLDigSig certificates, like the public key certificate information that are used during XMLDigSig verification process.

It was already stated that XMLDigSig can be used to sign specific elements, specific set of data. To specify the data that are going to be sign by XMLDigSig mechanism **<*Reference*>** element is used. It specifies the location of data or XML element that is signed.

XML SIGNATURE FORMAT (11, 12)

```
<Signature ID ?>
        <SignedInfo>
                <CanonicalizationMethod/>
                <SignatureMethod/>
                (<Reference URI ?>
                (<Transforms/>) ?
                <DigestMethod>
                <DigestValue/>
                </DigestMethod>
                </Reference >) +
        </SignedInfo>
        <SignatureValue/>
        (<KeyInfo/>) ?
        (<Object ID ? />) *
</Signature>
```

The mechanism of XMLDigSig is based on several steps:

1. To define the URI that will be signed;
2. To calculate the hash value;
3. To write the hash value into the **<*Reference*>** element, with additional information regarding the algorithms and other information;
4. To standardize the whole **<*SignedInfo*>** element, and calculate its hash and the signature value;
5. To write the signature value to the **<*SignatureValue*>** element;
6. To add certificate/ key information;
7. To combine **<*SignedInfo*>, <*SignatureValue*>** and **<*KeyInfo*>** into the **<*Signature*>** element.

XMLDigSig mechanism can be described on the following example. The goal is to sign 3 sets of data, which are stored on the folowing URL adresses:

"http://www.foi.hr/index.html" - HTML web page
"http://www.foi.hr/slika.jpg" – file in GIF format
"http://www.foi.hr/xml/popis.xml#JMBAG" – JMBAG field in popis.xml file

The first step is to calculate the hash values for specified content stored at *http://www.foi.hr/index.html* location. The adresses of content that will be signed are stored in **<*Reference*>** element, and calculated hash value in **<*DigestValue*>** element. Beside that in **<*DigestMethod Algorithm*>** element is written information about used algorithm (sha1 in this case).

```
<Reference URI="http://www.foi.hr">
 <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1
" />
<DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</
DigestValue>
</Reference>
```

The same is done for other two data sets, and then is all combined into **<*SignedInfo*>** element.

```
<SignedInfo Id="Primjer">
 <CanonicalizationMethod
   Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
 <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1" />
 < Reference URI="http://www.foi.hr/index.html">
  <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
<DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
 </Reference>
< Reference URI=" http://www.foi.hr/slika.jpg ">
  <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
<DigestValue>j6lwx3rvEPO0v123454NbeVu8nk=</DigestValue>
 </Reference>
 <Reference URI="
http://www.foi.hr/xml/popis.xml#JMBAG">
  <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>85214LBIta6skoV5qA8Q38GEw44=</DigestValue>
 </Reference>
</SignedInfo>
```

Based on digest values a digital signtaure is formed, and its value is stored into **<*SignatureValue*>** element.

```
<SignatureValue>PqLeN7E</SignatureValue>
```

All links and information regarding keys, certificates and verification process are stored into **<*KeyInfo*>** element.

```
<KeyInfo>
 <X509Data>
  <X509SubjectName>CN=Sandro Geric,O=FOI,ST=Varazdin,C=Croatia</X509SubjectName>
<X509Certificate>125478LkmNHJuzt...KLO9</X509Certificate>
 </X509Data>
</KeyInfo>
```

The final step is to combine all previously mentioned elements of XMLDigSig specification into root **<*Signature*>** element. The result is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Signature
xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo Id="Primjer">
 <CanonicalizationMethod
   Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
 <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1" />
 < Reference URI="http://www.foi.hr/index.html">
  <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
<DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
 </Reference>
< Reference URI=" http://www.foi.hr/slika.jpg ">
  <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
<DigestValue>j6lwx3rvEPO0v123454NbeVu8nk=</DigestValue>
 </Reference>
 <Reference URI="
http://www.foi.hr/xml/popis.xml#JMBAG">
  <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>85214LBIta6skoV5qA8Q38GEw44=</DigestValue>
 </Reference>
</SignedInfo>
<SignatureValue>PqLeN7E</SignatureValue>
<KeyInfo>
 <X509Data>
  <X509SubjectName>CN=Sandro Geric,O=FOI,ST=Varazdin,C=Croatia</X509SubjectName>
<X509Certificate>125478LkmNHJuzt...KLO9</X509Certificate>
 </X509Data>
</KeyInfo>
</Signature>
```

The verification of XMLDigSig is based on two main steps:

1. The signature confirmation – the receiver side is calculating their own hash values (digest) based on received content (signed data) and information about used algorithms. The result is calculated **<*SignatureValue*>** that is compared with received **<*SignatureValue*>** content. If they match, signature confirmation is positive.
2. The reference validation – comparing the calculated hash value with that stored in the **<*DigestValue*>** element. If they match then the original data set wasn't changed

Verification of the XML signature passes, only when all of the two steps above succeed.

V. IMPLEMENTATION OF XML DIGITAL SIGNATURE

An example of application modules that can be used for XMLDigSig implementation and verification is shown below (9).

1) Creating a signature

```
KeyStore keystore=KeyStore.getInstance("JKS");
keystore.load(new
FileInputStream(keystorepath),storepass)
X509Certificate cert=
(X509Certificate)keystore.getCertificate(usrdata);
Key key= keystore.getKey(usrdata,keypass);
If (key==null) {System.err.println("Could not get a
key,"+usrdata);System.exit();}
KeyInfo eyinfo=dsig.SignatureUtil.createKeyInfo(cert);
Element signatureElement=
signatureGen.getSignatureElement();
keyinfo.insertTo(signatureElement,prefix);
signatureContext sigContext=new SignatureContext();
sigContext.setIDResolver(sig);
sigContext.sign(signatureElement,key);
doc.appendchild(signatureElement);
dsig.SignatureUtil.printDocum(do,System.out);
```

2) Verification of signature

```
XmlDocument xdoc=new XmlDocument();
xdoc.PreserveWhitespace=true;
XmlTextReader xfile=new XmlTextReader(filename);
xdoc.Load(xfile);
xfile.Close();
SignedXml sx=new SignedXml(xdoc);
XmlNodeList nl=
xdoc.GetElementsByTagName("Signature",
"http://www.w3.org/2000/09/xmldsig#");
sx.LoadXml((XmlElement) nl[0]);
if (sx.CheckSignture())
{Console.Writeline("Sign check Pass !");}
else
{ Console.Writeline("Sign check Fail !");}
```

VI. XML ENCRYPTIONS

XMLDig Sig has a possibility to protect the confidentiality of documents as well. For that purpose XML encryption technology is used. It is a form of encryption mechanism optimized for encrypting XML data. Just like XMLDigSig it enable partial encryption which encrypts specific tags in XML structure, and multiple encryption which encrypts multiple tags in XML structure, or parts of XML document, or entire XML document. It has a possibility to define specific segments of encrypted data that can be decrypted by receiver, meaning that it is possible to determine which part of encrypted data a receiver can decrypt and which cannot. The use of XML Encryption prevents some security issues, like XML data eavesdropping, and it adds an additional level of functionality and protection to XMLDigSig.

XML Encription is defined by XML specification available at: http://www.w3.org/TR/xmlenc-core/#sec-Overview, and it defines following structure that is used for XML Encryption files.

XML Encryption format (11, 12)

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

The root element of XML Encryption is **<*EncryptedData*>** element that is used to store all information regarding encryption process, divided into previousely shown XML Schema elements. The **<*EncryptedData*>** element is the core element in the syntax. Not only does its **<*CipherData*>** child contain the encrypted data, but it's also the element that replaces the encrypted element, or serves as the new document root. **<*EncryptionMethod*>** is an optional element that describes the encryption algorithm applied to the cipher data. If the element is absent, the encryption algorithm must be known by the recipient or the decryption will fail. The **<*EncryptedKey*>** element is used to transport encryption keys from the originator to a known recipient(s). It may be used as a stand-alone XML document, be placed within an application document, or appear inside an **<*EncryptedData*>** element as a child of a **<*ds:KeyInfo*>** element. The key value is always encrypted to the recipient(s). When **<*EncryptedKey*>** is decrypted the resulting octets are made available to the **<*EncryptionMethod*>** algorithm without any additional processing. The **<*CipherData*>** is a mandatory element that provides the encrypted data. It must either contain the encrypted octet sequence as base64 encoded text of the **<*CipherValue*>** element, or provide a reference to an external location containing the encrypted octet sequence via the **<*CipherReference*>** element (1, 6).

The process of XML Encryption consists of several steps:

1. *to define an encryption algorithm that will be used* – XML encryption can use both symmetrical encryption and the asymmetrical encryption. Considering this for symmetrical algorithm DES and AES algorithms are mostly used, and RSA is mostly used in the domain of asymmetrical algorithms.

2. *to choose a key transmission method* – to ensure that both sides have a necessary encryption and decryption keys in case of symmetrical encryption a trusted third party is

used for key exchange. In case of asymmetrical encryption a public key infrastructure is used to distribute the necessary keys, and relations to them are ensured using X.509 certificates.

3. *to encrypt the primary data* – using defined encryption algorithms defined XML data are being encrypted. The results of encryption are stored into XML Encryption schema form.

VII. IMPLEMENTATION OF XML ENCRYPTION

An example of application modules that can be used for XML Encryption (encrypting and decrypting process) is shown below (1, 9).

1. Encryption process

First it is necessary to create a <***XmlDocument***> object, and load it into the XML file that will be encrypted:

```
XmlDocument xdoc=new XmlDocument();
xdoc.Load("encrypting.xml");
```

The next step is to create an <***EncryptedXml***> object, and transmit the object of previous step to it as a parameter:

```
EncryptedXml exml=new EncryptedXml(xdoc);
```

The function GetNumberingKey() will provide us with key:

```
RSA numberingKey=GetNumberingKey();
exml.AddKeyNameMapping("id",
numberingKey);
```

The element that will be encrypted is being captured, and data are being encrypted using encrypt method:

```
XmlNodeList xnode=
xdoc.GetElementsByTagName('cardid');
XmlElement xelement = XmlElement(xnode[0]);
EncryptedData encryptedNeedEncrypt =
exml.Encrypt(xelement,"id");
```

Finally, it is necessary to replace the unencrypted part of the original XML file with the new encrypted data:

```
exml.ReplaceElement(xelement,encryptedNeedEn
crypt,true);
```

2) Decryption process

First it is necessary to load the encrypted XML file with necessary keys and information:

```
XmlDocument xdoc = new
XmlDocument("encrypted.xml");
EncryptedXml exml = new
EncryptedXml(xdoc,documentEvidence);
RSA numberingKey = GetNumberingKey();
exml.AddKeyNameMapping("numbering",
numberingKey);
```

The second step is to decrypt the file using decrypt method:

```
exml.DecryptDocument();
```

VIII. CONCLUSION

XML signature is form of digital signature designed for use in XML transactions. The XML Digital Signature standard defines a schema that is used for storing the result of a digital signature operation applied to (in most cases) XML data. Like non-XML digital signatures, XML signatures add authentication, data integrity, and support for non-repudiation to the data that is object of XML digital signing process. However, unlike non-XML digital signature standards, XML signature has been designed to both account for and take advantage of the Internet and XML. A fundamental feature of XML Signature is the ability to sign only specific portions of the XML content rather than the complete document. This is relevant when a single XML document may have a long history in which the different components are authored at different times by different parties, each signing only those elements relevant to it. This flexibility will also be critical in situations where it is important to ensure the integrity of certain portions of an XML document, while leaving open the possibility for other portions of the document to change. Since data security – in form of data verification and authorization - represents an important part of information system security paradigm this article is addressing the questions and possibilities of XMLDigSig usage in everyday information system security procedures. Regarding all this XMLDigSig presents a valuable mechanism in security of electronic transactions that are mostly based on exchange of XML data.

LITERATURE

[1.] Ahmed, S., Armstrong, L., Securing Web Services with XML aware digital signatures, http://citeseerx.ist.psu.edu, 02.12.2011.

[2.] Alhir, S., The Object-Oriented Paradigm, 1998.

[3.] Champion, M., Ferris, C., Newcomer, E.,Orchard, D. Web Services Architecture, 2002. http://www.w3.org/TR/2002/ WD-ws-arch-20021114/, 03.11.2011.

[4.] Gerić, S., Vrček, N, Divjak, L., Modern Information Systems Architectures and their Impact on Organizations, International Academic Conference Social Technologies 2011: ICT for Social Transformations, Mykolas Romeris University, Vilnius, Litva, 2011, pp. 33 - 38.

[5.] Gerić, S., Security of Web Services Based Service-Oriented Architectures, MIPRO 33rd International Convention, Proceedings of Information System Security, Opatija, 2010, pp. 208 - 213.

[6.] Gu, Y., Ye, M., Web Services Security Based on XML Signature, JOURNAL OF NETWORKS, VOL. 5, NO. 9, 2010.

[7.] Simon, E.; Madsen, P. & Adams, C., An Introduction to XML Digital Signatures, http://www.xml.com/pub/ a/2001/08/08/xmldsig.html, 12.11.2011.

[8.] Slama, D. et al: Service Oriented Architecture: Inventory of Distributed Computing Concepts, Prentice Hall PTR, 2004.

[9.] Snell, J., Tidwell, D., Kulchenko, P, Programming Web Services with SOAP, O'Reilly Media, 2001

[10.] Szyperski, C. (1998): Component Software: Beyond Object-Oriented Programming, Addision-Wesley

[11.] ***: XML Encryption Syntax and Processing, http://www.w3.org/TR/xmlenc-core/#sec-EncryptedKey, 12.09.2011.

[12.] ***: XML Web services fundamentals, https://www6.software.ibm.com/developerworks/education/ws-intwsdk51/ws-intwsdk51-2-1.html, 12.09.2011.