Representation of E-documents in AIDA Project

Diana Berbecaru Marius Marian

Dip. di Automatica e Informatica
Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129 Torino, Italy

Abstract

Initially developed and considered for providing authentication and integrity functions, digital signatures are studied nowadays in relation to electronic documents (e-docs) so that they can be considered equivalent to handwritten signatures applied on paper documents. Nevertheless a standardized format to be used specifically for e-doc representation was not yet indicated. Each system is free to choose whatever e-doc format is suitable for their needs (e.g. ASCII, Word, PDF, binary). So far, there has been very little work on design and implementation of a *secure* document management system that enables digital signing and easy management of e-docs. A solution to this problem is our document management system named AIDA. This paper explains in detail the use of XML and XML Signature for the representation of e-docs in AIDA. Extended details on the overall system's functionality and practical use can be found in [2, 3, 4].

Keywords: digital signature, e-doc, XML Signature

1. INTRODUCTION

Public key cryptography was introduced in 1976 by Diffie & Hellman and makes use of a pair of mathematically related keys used for encryption and decryption; one key, named the private key is only known by the owner, while the other one, named the public key is publicly known. To securely bind a public key to an entity, a data structure named *public-key certificate* (PKC) is used.

One open issue in applying PKC in real world scenarios is the use of digital signatures in relation to e-docs so that they can be considered equivalent to handwritten signatures applied on paper documents. The EC-funded AIDA project (IST-1999-10497) [1] provides a secure document management architecture, which is based on standards and technologies available on the market today and cutting edge techniques, such as smart-cards and handheld PCs. A detailed description of the AIDA architecture and the workflows implemented are found in [2, 3, 4].

This paper is focused only on the representation of e-docs in AIDA. The paper is structured as follows: Section 2 explains the main types of cryptographic envelopes used to group altogether the data to be signed, the digital signatures applied on the data and the signer's digital certificate, Section 3 presents the formats of e-docs in AIDA and Section 4 explains how to define the content of an example AIDA e-doc that expresses a tax declaration. For simplicity, we'll avoid the complete description of all AIDA components and their functionality. We'll keep in mind that the e-docs are stored on a dedicated component on the AIDA server, called *definitions repository* and a dedicated tool named *Definitions Manager* is used for creating the data structures that describe the content of e-docs for a specific workflow. These data structures are called AIDA document type definitions and, because are digitally signed, are e-docs themselves.

2. FORMATS FOR DIGITAL SIGNATURE

2.1 Definition of digital signature

A *digital signature* is a mark that only the sender can make, but other people can easily recognize as belonging to the sender [5]. A digital signature is used to confirm agreement to a message, just like a real signature does. A digital signature must meet two primary conditions:

1. *unforgeable*. If sender Alice signs message M with signature S(Alice,M), it is impossible for anyone else to produce the pair [M,S(Alice,M)].
2. *authentic.* If a receiver Bob receives the pair [M,S(Alice,M)], purportedly from Alice, Bob can check that the signature is really from Alice. Only Alice could have created this signature, and the signature is firmly attached to M.

In a public key cryptosystem, Message (M) is signed by appending to it an enciphered summary of the information. The summary is produced by means of a one-way hash function h(), while the enciphering is carried out using the secret key of the signer. Thus, the digital signature performed by Alice on the message M is expressed as:

$$S(Alice,M) = Encrypt(PvKAlice, h(M))$$

The encipherment using the private key ensures that the signature cannot be forged. The one-way nature of the hash function ensures that false information, generated so as to have the same hash result (and thus signature), cannot be substituted. The receiver Bob of signed information [M, S(Alice,M)] verifies the Alice's signature by applying first the one-way hash function to the information, h(M) and comparing afterwards the result from the above point with that obtained by deciphering the signature using the public key of the signer,

$$h(M) = Decrypt(S(Alice,M), PbK_{Alice})$$

The cryptographic envelopes were defined to express the relation among the data to be signed and the digital signature. In this sense we can distinguish: *enveloping* signatures when the data content in inserted in the digital signature envelope; *enveloped* signatures when the signature is inserted in the data content that need to be signed; *detached* signatures when the digital signature object is completely separated from the data content that need to be signed.

For the *detached* signatures it is not specified any way to establish a strict correlation between the data that was signed and the digital signature object. For this reason it is required to have an external mechanism to maintain the correspondence between them, like for example to store them in the same database and to link them in some way. This kind of problem is solved in the *enveloping* signature. The digital signature is bound to the data by mean of some syntax that specifies the format of digital signature like for example PKCS#7/CMS [6]. Any number of signers in parallel uses the *signed-data* content type when it is the need to add a digital signature to arbitrary data content. Format for data to be signed is blob (block of bits). The syntax supports also *detached* signatures that is the data and the corresponding signatures are not grouped together in a whole message of *signed-data* content type; in this case the mechanism for the verification of the signatures is application dependent. In *enveloped* signature instead, the digital signature is part of the document itself. For example, Adobe Acrobat supports signatures embedded into data in Portable Document Format (PDF).

2.2 XML Signature

XML (eXtensible Markup Language) is a meta-language, i.e. a language that allows defining other languages. XML is part of a more complex language named Standard Generalized Markup Language (SGML) defined in the ISO standard (8879:1986). The language defined for expressing digital signatures is XML Signature. The most important element defined in the XML Signature standard is the *Signature* element.

Since XML is used in AIDA to express the e-doc we will shortly sketch the main parts of the *Signature* element here. As known from the paragraph above on PKCS#7/CMS, the hash value of the data to be signed is not actually signed directly. Rather the hash values of all data elements that should be signed are listed, and then the hash value of this list is signed. It is a kind of double indirection. The compulsory *SignedInfo* element (lines 02 to 10 in Table 1) holds this list of references that need to be signed. Each reference stored in the *Reference* element holds a reference identifier, which is normally a URI (attribute in the element *Referenc*e), the method used for the computation of the hash (the element *DigestMethod*) and a hash value of the referenced data (the element *Digest Valu*e). When we say "reference to an element", this refers to an element with all its XML element attributes and all descendant elements; this is the whole sub-tree with the denoted element as its root.

The *KeyInfo* element (lines 12 to 14 in Table 1) holds the signing certificate or a reference to it. This element indicates the key that must be used to validate the signature. Possible forms of identification of this data include certificates, key names, and algorithms for the key exchange. Because the *KeyInfo* element is placed outside the *SignerInfo* element, if the signer wants to bind to the signature the information related to the key, it is possible to use a *Reference* element that can easily identify and include *KeyInfo* as part of the signature itself. The validation of the *SignedInfo* consists of two obligatory processes: the validation of the signature applied on the *SignedInfo* structure and the validation of each hash of the *Reference* structures from the *SignedInf*o. The calculated signature value is in the *SignatureValue* element as base-64 encoded binary data. We can note also that the algorithms used in the calculation of the *SignatureValue* are included in the signed data, while the element *SignatureValue* is external to the element *SignedInfo*.

```
<Signature Id="ExampleXMLSignature"
xmlns="http://www.w3.org/2000/07/xmldsig#">
<SignedInfo >
<CanonicalizationMethod Algorithm= …/>
<SignatureMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<Reference URI="http://www.w3.org/…">
 <Transforms> <Transform …/> </Transforms>
 <DigestMethod Algorithm=
"http://www.w3.org/2000/07/xmldsig#sha1"/>
 <DigestValue> j6lwx3rvEP=…< /DigestValue >
</ Reference>
</SignedInfo>
<SignatureValue> MC0CFF=...< /SignatureValue >
<KeyInfo>
 <KeyValue> … </ KeyValue>
</ KeyInfo>
</Signature>
```

Table 1: Example of *Signature* element

XML Signature can be used to sign entire XML documents or parts of them or any binary data. XML Signature offers support to *enveloped signature* only for XML data, and to *enveloping* and *detached* signatures both for XML documents and for generic blobs.

3. E-DOC REPRESENTATION IN AIDA

We have stated that e-docs are in general signed documents in electronic form. An example of such a widely used signed document format is signed email, which uses the PKCS#7 format to encode its data. The basic format of all formats is roughly the same. A signed electronic document consists of the following parts:

1. the signed content;
2. one or more independent *parallel* signatures, i.e. the signatures applied on the same content. Each signature has *signed* and *unsigned* attributes.

We'll describe further how XML was used in AIDA to encode e-docs. A detailed explanation can be found in [9]. In AIDA the e-doc is a well-formed XML document. The basic structure is represented as a tree where the root element of the document is *eDocument*. The first child element is the *signedContent* element, which holds the content data that should be signed. The content can be an *eDocument* once again or it can be any other well-formed XML document. Following the *signedContent*, the root element has one or more *Signature* elements. The structure of the *Signature* element and its children is compliant to the XML Signature format. The *SignedInfo* element of the *Signature* element contains references to:

1. the *signedContent* element, which contains the content data to be signed;
2. the *KeyInfo* element holds the signing certificate or a reference to it;
3. *signedAttributes*.

The *Signature*'s Object element contains *signed* and *unsigned attributes*. The *unsignedAttributes* element can hold any attributes that cannot or should not be covered by the signature value. Normally, this is used for attributes that are not available at the creation time of the signature. Typical unsigned attributes are timestamps, revocation information like CRLs or additional certificates. The structure of the *signedAttributes* element will be defined in the upcoming document of the ETSI for XML Electronic Signatures. This document will define similar structures for XML as already defined for CMS in the ETSI document Electronic Signature Formats [8].

4. EXAMPLE E-DOCUMENT IN AIDA

This section shows an example e-doc representing a sample tax declaration. To implement a workflow that uses tax declaration e-docs there must be defined first the document type definitions (DTDs) and the display transformations. A DTD contains the e-doc's content while the display transformation contains the XML transforms (or stylesheets) used to present the e-docs content to the user. To create and sign the DTDs and the transformations, the *Definitions Manager* tool is used. In AIDA, all DTDs and transforms stored in the *definitions repository* must have a unique ID, which is assigned at the definition time. A complete DTD contains a document type ID expressed by the <*aida:documentTypeID*> element and the actual XML Schema contained in the element <*aida:schema*>. The document type ID is up to 100 characters long and looks like a URL, such as "aida://www.polito.it/tax". For the definition of the content of an e-doc, the AIDA system allows to define:

1. either a *generic* type definition. Anyone can easily create such a generic schema by defining only the name and types of the fields, without knowing details of XML Schema. The generic schema is expressed by the element <*aida:genericSchema*>. A DTD of this kind is processed further with the Definitions Manager tool, which will ask the definer for the preferred document type ID. The tool will create automatically the XML Schema and the enveloping AIDA e-doc.

2. or a *specific* type definition. This can be any correct XML Schema definition.

The generic DTD for tax declaration is shown in Table 2.

```
<aida:genericSchema
xmlns:aida="http://aida.infonova.at">
<aida:documentRoot> tax
</aida:documentRoot>
<aida:documentNamespace>
http://www.polito.it/tax
</aida:documentNamespace>
<aida:namespacePrefix> polito
</aida:namespacePrefix>
<aida:fieldList>
<aida:field>
   <aida:name> Unique_identification_number
   </aida:name>
<aida:shortString max="20" searchable="true"/>
</aida:field>
<aida:field>
    <aida:name>Surname</aida:name>
  <aida:shortString max="20" searchable="true"/>
</aida:field>
<aida:field>
    <aida:name>Name</aida:name>
  <aida:shortString max="20" searchable="true"/>
</aida:field>
<aida:field>
    <aida:name> Income_from_buildings_fields
    </aida:name>
    <aida:shortString max="70"/>
</aida:field>
<aida:field>
    <aida:name> Income_as_employee </aida:name>
    <aida:shortString max="70"/>
</aida:field>
<aida:field>
    <aida:name>Other_incomes</aida:name>
    <aida:shortString max="80"/>
</aida:field>
<aida:field>
    <aida:name>Taxes_Expenses</aida:name>
    <aida:shortString max="80"/>
</aida:field>
<aida:field>
    <aida:name>Phone_number</aida:name>
    <aida:shortString max="30"/>
</aida:field>
<aida:field>
    <aida:name>Mail_address</aida:name>
    <aida:shortString max="200"/>
</aida:field>
</aida:fieldList>
</aida:genericSchema>
```

Table 2. Generic type definition for tax declaration

The generic type definition contains:

1. the name of the e-document XML root element contained in the tag <*aida:documentRoot*>

2. the XML namespace identifier contained in the element <*aida:documentNamespace*>

3. the XML namespace prefix to be used for each field element contained in the element <*aida:namespacePrefix*>

4. the actual list of fields of the e-doc

All the XML tags are prefixed with the *aida:* namespace prefix and the *xmlns:aida="http://aida.infonova.at"* XML namespace identifier is used in the root element. The <*aida:fieldList*> tag can contain an arbitrary number of <*aida:field*> tags. Each <*aida:field*> tag must contain one <*aida:name*> tag with the desired field name and one empty tag denoting the field type. The field types can take one of the values shown in Table 3.

Name	XML Schema datatype	Comment
aida:string	string	no length limit
aida:shortString	string	maximum 250 characters
aida:date	date	fixed format YYYY-MM-DD of XML element value representation
aida:time	time	fixed format HH:MM:SS.SSS of XML element value representation
aida:int	int	signed 32-bit integer in canonical textual form
aida:double	double	IEEE standard double 64-bit in canonical textual form
aida:boolean	boolean	represented as "true" or "false"

Table 3. Simple AIDA field types

Using the generic type definition for tax declaration the e-administrator can build the AIDA DTD for the tax declaration having <*aida:documentTypeData*> as root element. The AIDA DTD can be further exported in an XML file having the content illustrated in the Table 4. Such DTDs are constructed from generic or specific type definition input files with the *Definitions Manager* tool but they can also be constructed by hand since the structure of the enveloping <*aida:documentTypeData*> element is not complex.

```
<?xml version="1.0" encoding="UTF-8" ?>
<aida:documentTypeData
xmlns:aida="http://aida.infonova.at"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-insta
nce xsi:schemaLocation="http://aida.infonova.at
aida:documentTypeData">
<aida:documentTypeID>
   aida://www.polito.it/tax
</aida:documentTypeID>
<aida:schema>
  <xsd:schema
targetNamespace="http://www.polito.it/tax"
xmlns:polito="http://www.polito.it/tax"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="tax">
 <xsd:complexType>
   <xsd:sequence>
     <xsd:element
ref="polito:Unique_identification_number" />
     <xsd:element ref="polito:Surname" />
     <xsd:element ref="polito:Name" />
     <xsd:element
ref="polito:Income_from_buildings_fields" />
  <xsd:element ref="polito:Income_as_employee" />
     <xsd:element ref="polito:Other_incomes" />
```

```
      <xsd:element ref="polito:Taxes_Expenses" />
      <xsd:element ref="polito:Phone_number" />
      <xsd:element ref="polito:Mail_address" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:simpleType name="shortString">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="250" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:element name="Unique_identification_number">
  <xsd:simpleType>
     <xsd:restriction base="polito:shortString">
     <xsd:maxLength value="20" />
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:element>
  ...
 </xsd:schema>
</aida:schema>
<aida:genericSchema
xmlns:aida="http://aida.infonova.at">
<aida:documentRoot>tax</aida:documentRoot>
<aida:documentNamespace> http://www.polito.it/tax
</aida:documentNamespace>
<aida:namespacePrefix> polito
</aida:namespacePrefix>
 <aida:fieldList>
   <aida:field>
     <aida:name> Unique_identification_number
     </aida:name>
     <aida:shortString max="20"
      searchable="true" />
   </aida:field>
   ...
 </aida:fieldList>
 </aida:genericSchema>
</aida:documentTypeData>
```

Table 4. AIDA type definition built from generic schema

To be stored in the definitions repository, the AIDA DTD is further signed with the Definitions Manager. The e-doc containing the signed AIDA document type definition is illustrated in Table 5. We can note that the signed AIDA type definition is an e-doc itself as it is contained in the element <*aida:eDocument*> and has the format described in Section 3. The signature is applied on the element <*aida:signedContent*> that contains at its turn the element <*aida:documentTypeData*> and its child elements that were shown in Table 4. The XML signature element is contained in the element <*dsig:Signature*> .

```
<?xml version="1.0" encoding="UTF-8"?>
<aida:eDocument
xmlns:aida="http://aida.infonova.at"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instan
ce xsi:schemaLocation="http://aida.infonova.at
aida:eDocument">
<aida:signedContent>
<aida:documentTypeData
xmlns:aida="http://aida.infonova.at"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instan
ce xsi:schemaLocation="http://aida.infonova.at
aida:documentTypeData">
 <aida:documentTypeID> aida://www.polito.it/tax
 </aida:documentTypeID>
 <aida:schema>
  <xsd:schema
targetNamespace="http://www.polito.it/tax"
xmlns:polito="http://www.polito.it/tax"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="tax">
   <xsd:complexType>
    <xsd:sequence>
     <xsd:element
ref="polito:Unique_identification_number"/>
     <xsd:element ref="polito:Surname"/>
     <xsd:element ref="polito:Name"/>
     <xsd:element
ref="polito:Income_from_buildings_fields"/>
     <xsd:element
ref="polito:Income_as_employee"/>
     <xsd:element ref="polito:Other_incomes"/>
     <xsd:element ref="polito:Taxes_Expenses"/>
     <xsd:element ref="polito:Phone_number"/>
     <xsd:element ref="polito:Mail_address"/>
    </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
<xsd:simpleType name="shortString">
  <xsd:restriction base="xsd:string">
   <xsd:maxLength value="250"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:element name="Unique_identification_number">
  <xsd:simpleType>
  <xsd:restriction base="polito:shortString">
    <xsd:maxLength value="20"/>
  </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
...
</xsd:schema>
</aida:schema>
<aida:genericSchema>
  <aida:documentRoot>tax</aida:documentRoot>
<aida:documentNamespace> http://www.polito.it/tax
  </aida:documentNamespace>
  <aida:namespacePrefix> polito
  </aida:namespacePrefix>
<aida:fieldList>
  <aida:field>
    <aida:name>Unique_identification_number
    </aida:name>
   <aida:shortString max="20" searchable="true"/>
  </aida:field>
  ...
  </aida:fieldList>
 </aida:genericSchema>
</aida:documentTypeData>
</aida:signedContent>
<dsig:Signature
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
<dsig:SignedInfo>
<dsig:CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml
-c14n-20010315"/>
<dsig:SignatureMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<dsig:Reference
URI="#xmlns(aida=http://aida.infonova.at)… ">
<dsig:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1
"/>
<dsig:DigestValue> 8/E+Xcs=… </dsig:DigestValue>
</dsig:Reference>
<dsig:Reference
URI="#xmlns(aida=http://aida.infonova.at)… ">
<dsig:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1
"/>
<dsig:DigestValue>1Gz1wk43I= </dsig:DigestValue>
</dsig:Reference>
<dsig:Reference
URI="#xmlns(aida=http://aida.infonova.at)… ">
<dsig:DigestMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/>
```

```
<dsig:DigestValue>onIowJD39hCKYFuHCUroaRKXPPE=</d
sig:DigestValue>
</dsig:Reference>
</dsig:SignedInfo>
<dsig:SignatureValue> JZLLunRKQFxaw+GlS …
</dsig:SignatureValue>
<dsig:KeyInfo> <dsig:X509Data>
   <dsig:X509Certificate> jCCAjegAwIBAgIBATAJBg…
   </dsig:X509Certificate>
  </dsig:X509Data> </dsig:KeyInfo>
<dsig:Object>
  <aida:properties
xmlns:aida="http://aida.infonova.at">
   <aida:signedProperties/>
   <aida:unsignedProperties/>
  </aida:properties>
</dsig:Object>
</dsig:Signature>
</aida:eDocument>
```

Table 5. Signed AIDA document type definition defining the content for example tax declaration e-doc

An example tax declaration e-document instance is illustrated in Table 6.

```
<?xml version="1.0" encoding="UTF-8"?>
<aida:eDocument xmlns:aida=
http://aida.infonova.at xmlns:xsi=
http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation= "http://aida.infonova.at
aida:eDocument">
<aida:signedContent>
  <polito:tax
xmlns:polito="http://www.polito.it/tax"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-insta
nce xsi:schemaLocation="http://www.polito.it/tax
aida://www.polito.it/tax">
<polito:Unique_identification_number>D12876
  </polito:Unique_identification_number>
  <polito:Surname>Popescu</polito:Surname>
  <polito:Name>Ion</polito:Name>
  <polito:Income_from_buildings_fields>
    21,000,000
  </polito:Income_from_buildings_fields>
  <polito:Income_as_employee> 2,000,000
  </polito:Income_as_employee>
  <polito:Other_incomes> 1,000,000
  </polito:Other_incomes>
  <polito:Taxes_Expenses> 2,000,000
  </polito:Taxes_Expenses>
  <polito:Phone_number> +22323214
  </polito:Phone_number>
  <polito:Mail_address> Popescu.Ion@domain.com
  </polito:Mail_address>
 </polito:tax>
</aida:signedContent>
<dsig:Signature
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
<dsig:SignedInfo>
 <dsig:CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14
n-20010315"/>
<dsig:SignatureMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<dsig:Reference
URI="#xmlns(aida=http://aida.infonova.at)… ">
<dsig:DigestMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/>
<dsig:DigestValue>k+nbo…=</dsig:DigestValue>
</dsig:Reference>
<dsig:Reference URI=
"#xmlns(aida=http://aida.infonova.at)… ">
<dsig:DigestMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/>
<dsig:DigestValue>J1KMyLXteg=…/dsig:DigestValue>
</dsig:Reference>
<dsig:Reference URI=
"#xmlns(aida=http://aida.infonova.at)… ">
<dsig:DigestMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/>
<dsig:DigestValue>hDl7B/HNk=…</dsig:DigestValue>
 </dsig:Reference>
</dsig:SignedInfo>
<dsig:SignatureValue> Ma/mMmYGC5t1fDNtfU6BymvR…
</dsig:SignatureValue>
<dsig:KeyInfo> <dsig:X509Data>
    <dsig:X509Certificate> MIIC8TCCAl6gAwIB…
    </dsig:X509Certificate>
  </dsig:X509Data> </dsig:KeyInfo>
<dsig:Object>
 <aida:properties
xmlns:aida="http://aida.infonova.at">
  <aida:signedProperties>
    <aida:transformDataID>taxTrafo1
    </aida:transformDataID>
    <aida:documentHash> tVzJfpiBHZrJN=…
    </aida:documentHash>
  </aida:signedProperties>
  <aida:unsignedProperties>
    <aida:signatureValueTimeStamp>
      MIIULjADAgEAMIIUJQYJK…
    </aida:signatureValueTimeStamp>
  </aida:unsignedProperties>
 </aida:properties>
</dsig:Object>
</dsig:Signature>
</aida:eDocument>
```

Table 6. AIDA e-doc instance for tax declaration

5. CONCLUSIONS

AIDA Project implemented a secure XML-based document management system that can be easily adapted to different types of real world scenarios where e-docs must be digitally signed. One of the main requirements in the design phase was to analyse and choose a format for the digital signature and to design and implement the format of the e-doc. This paper describes the most widely used formats for signature cryptographic envelopes and describes the representation of e-doc in AIDA based on XML and XML Signature.

Reference

1. Advanced Interactive Digital Administration (AIDA) Project, http://aida.infonova.at.
2. D. Berbecaru, A. Lioy, M. Marian. *A Framework for Secure Digital Administration*, Proc. of EuroWeb2001 Conference, pp: 119 - 133, 2001.
3. D. Berbecaru, A. Lioy, D. Marano, M. Marian. *Secure Digital Administration in Medical Environment*, Proc. of IADIS International Conference WWW/Internet 2002, pp: 299-306, 2002.
4. D. Berbecaru, *Architectures and protocols for practical use of public key cryptography, pp: 6-77*. Ph. D. Thesis, 2003.
5. C.P. Pfleeger, *Security in Computing*, Prentice Hall, 1997.
6. R. Housley, *Cryptographic Message Syntax*. IETF, RFC-2630, 1999.
7. R. Housley, W. Polk, W. Ford, D. Solo, *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. IETF, RFC-3280, 2002.
8. Electronic Telecommunications Standards Institute, *Electronic Signature Formats, ETSI TS 101 733 V1.3.1*. 2002.
9. K. Scheilbelhofer, *Signing XML documents and the concept of What You See Is What You Sign*, Masters Thesis in Telematics, Institute for Applied Information Processing and Communications at Graz University of Technology, 2001.