Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Département fédéral des finances DFF

Office fédéral de l'informatique et de la télécommunication OFIT
Centre de solutions
Solutions Berne
Développement 1

Matthias Rüedlinger

Contrôle des signatures XML e-dec

Recommandations pour la mise en œuvre du contrôle des signatures numériques (WS-Security)

Nom du projet:	**e-dec**
Version:	**0.3**
Date:	**2009-05-27**

Statut	en travaux	à l'examen	approuvé pour utilisation
	☐	☐	☑

Personnes concernées	
Auteurs:	Matthias Rüedlinger (mru)
Approbation:	Equipe de projet e-dec IDEE
Utilisateurs:	AFD, direction du projet
Pour information / pour prendre connaissance:	AFD

Contrôle des modifications, examen, approbation			
Quand	**Version**	**Qui**	**Description**
2009-04-24	0.1	mru	Première version
2009-05-14	0.2	mru	Ajouté exemple de code Java
2009-05-15	0.3	mru, shu	Adaptations dans le chapitre CRL, adaptation du titre

Table des matières

1 Introduction

Le présent document s'adresse aux clients de la douane utilisant e-dec et aux fournisseurs de logiciels qui désirent vérifier la signature XML d'une décision de taxation électronique (DTE).

Le document décrit la manière dont les normes décrites dans la description des interfaces [1] et dans le contrat de service [2] pour l'EdecReceiptService peuvent être mises en œuvre.

Le contrôle de la signature de documents XML a été spécifié par l'organisation W3C. La spécification correspondante, appelée **XML Signature Syntax and Processing (XMLDsig)**, figure à l'adresse suivante:

http://www.w3.org/TR/xmldsig-core/.

Pour pouvoir effectuer le contrôle de la signature d'un document XML, la langue de programmation correspondante ou le *framework* du standard W3C **XML Signature Syntax and Processing (XMLDsig)** doit être mis en œuvre.

La signature XML est insérée dans une enveloppe SOAP. L'ancrage de la signature dans l'en-tête SOAP est effectué d'après le standard de sécurité WS.

1.1 Remarque

Les exemples de codes mentionnés ont simplement pour but de montrer comment on contrôle la signature et la chaîne de confiance du certificat dans une réponse DTE. Le code n'a pas été optimisé en matière de performance.

1.2 Références

Les documents suivants contiennent des informations sur la signature numérique dans e-dec.

Réf.	Titre	Version
[1]	Description des interfaces e-dec décision de taxation (description des messages à l'entrée et à la sortie pour le service)	1.5
[2]	Contrat de service EdecReceiptService (description des canaux de communication – service Internet et courrier électronique)	1.3

2 Outils / *frameworks*

La liste suivante est une sélection d'outils et de *frameworks* que l'on peut utiliser pour le contrôle de la signature XML:

Outil / *framework*	**Description**	**URL**
Java SE 6	Dans Java SE 6, on a la possibilité de vérifier avec les signatures API XML fournies en même temps.	http://java.sun.com/javase/
Apache XML Security	Dans Apache XML Security, on a une bibliothèque à l'aide de laquelle on peut vérifier les signatures XML pour Java ou C++.	http://santuario.apache.org/
IAIK XML Security Toolkit (XSECT)	Il s'agit d'une bibliothèque Java commerciale pour le contrôle de la signature XML.	http://jce.iaik.tugraz.at

3 Contrôle de la signature XML avec Java

Le mieux est d'utiliser la signature numérique XML API (JSR 105), qui est spécifiée par le Java Community Process (JCP). Voici la liste de quelques applications:

- Depuis Java 6, la signature numérique XML API est intégrée dans l'édition Java standard.
- Apache XML Security est une application libre de la signature numérique XML API.
- L'IAIK XML Security Toolkit (XSECT) est une application commerciale de la signature numérique XML API.

3.1 Signature numérique XML API

L'exemple suivant utilise l'interface standard de la signature numérique XML API; l'application pour laquelle on se décide joue donc un grand rôle. Ce n'est que lors de l'enregistrement du fournisseur de sécurité qu'il existe quelques différences.

Remarque: dans cet exemple, la CRL (*certification revocation list*, liste des certificats de sécurité révoqués) de la PKI (*public key infrastructure*, infrastructure à clé publique) Admin n'est pas encore contrôlée.

3.1.1 Exemple: contrôler la signature XML

On commence par lire le document XML. Il est important que le DocumentBuilder soit **namespace aware**.

```
InputStream is = XMLDsigClient.class.getResourceAsStream("eVVResponse.xml");

// create DocumentBuilderFactory which is Namespace aware
DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
builderFactory.setNamespaceAware(true);

DocumentBuilder builder = builderFactory.newDocumentBuilder();
logger.info("Is DocumentBuilder NamespaceAware: " + builder.isNamespaceAware());

// parse xml file (dumped soap request)
Document xmldoc = builder.parse(is);
```

XPath doit être initialisé avec un **NamespaceContext** propre, afin que l'on puisse utiliser l'espace nominal (*Namespace*) lors d'une interrogation XPath. Le **NamespaceContextImpl** met en œuvre le NamespaceContext de l'interface et doit être établi par l'utilisateur. (Voir exemple de NamespaceContext)

```
// create XPath object which has own NamespaceContext
XPath xpath = XPathFactory.newInstance().newXPath();
// with a custom NamespaceContext we can use Namespaces in our XPath query
NamespaceContext nsc = new NamespaceContextImpl();
xpath.setNamespaceContext(nsc);
```

Avec XPath, on peut extraire le **jeton X509 (X509 Token)**. Ce jeton X509 contient un certificat X509 qui est encodé en Base64.

```
// extract x509 Token --> xml element wsse:BinarySecurityToken
XPathExpression expr = xpath.compile("//wsse:Security/wsse:BinarySecurityToken");
Node x509Node = (Node) expr.evaluate(xmldoc, XPathConstants.NODE);
```

Nous pouvons utiliser les données du jeton X509 pour établir un certificat X509. Cependant, dans le certificat, il faut encore marquer le début par **-----BEGIN CERTIFICATE-----** et la fin par **-----END CERTIFICATE-----**. Sinon, on ne peut pas lire le certificat.

```
// X509 Token is encoded in base64
String header = "-----BEGIN CERTIFICATE-----\n";
String footer = "\n-----END CERTIFICATE-----";
// so we need a to add the certificate header and footer
// to the raw x509 data
byte[] x509data = (header + x509Node.getTextContent() + footer).getBytes();
ByteArrayInputStream bis = new ByteArrayInputStream(x509data);

// create certificate
CertificateFactory cf = CertificateFactory.getInstance("X.509");
X509Certificate cert = (X509Certificate) cf.generateCertificate(bis);
```

Ici, le certificat X509 et la chaîne de confiance sont vérifiés à l'aide du certificat CA. Si le certificat ou la chaîne de confiance ne sont pas valables, une exception est soulevée par le **CertPathValidator.validate(…)**.

```
// verify ca chain
// read in ca cert
is = XMLDsigClient.class.getResourceAsStream("adminca-cd-t01_BIT_CA_certificate.crt");
X509Certificate caCert = (X509Certificate) cf.generateCertificate(is);

// trusted ca cert
Set<TrustAnchor> trust = Collections.singleton(new TrustAnchor(caCert, null));
PKIXParameters params = new PKIXParameters(trust);

// Disable CRL checking since we are not supplying any CRLs
params.setRevocationEnabled(false);
// sets the time for which the validity of the certification
// path should be determined
params.setDate(new Date());

CertPath certPath = cf.generateCertPath(Collections.singletonList(cert));
CertPathValidator certPathValidator = CertPathValidator.getInstance("PKIX");
PKIXCertPathValidatorResult result = (PKIXCertPathValidatorResult) certPathValidator
    .validate(certPath, params);
```

L'élément de signature XML est extrait au moyen de XPath, et un JSR 105 Provider est initialisé. Ici, le provider est initialisé de façon explicite; cela est par exemple nécessaire si l'on travaille avec Apache XML Security. Avec Java 6, un provider de signature numérique XML API (JSR 105) est déjà enregistré, et cette étape ne devrait pas être nécessaire.

```
// extract xml element ds:Signature
expr = xpath.compile("//wsse:Security/ds:Signature");
Node dsSignature = (Node) expr.evaluate(xmldoc, XPathConstants.NODE);

DOMValidateContext context = new DOMValidateContext(cert.getPublicKey(), dsSignature);

String providerName = System.getProperty("jsr105Provider",
    "org.jcp.xml.dsig.internal.dom.XMLDSigRI");

logger.info("jsr 105 provider: " + providerName);

XMLSignatureFactory factory = XMLSignatureFactory.getInstance("DOM", (Provider) Class
    .forName(providerName).newInstance());

XMLSignature signature = factory.unmarshalXMLSignature(context);
```

La signature XML est contrôlée avec le **DOMValidateContext**. Celui-ci possède la clé publique et une référence à l'élément de signature XML.

```
      // Check core validation status
      boolean coreValidity = signature.validate(context);

      if (coreValidity == false) {

        logger.error("Signature failed core validation!");
        boolean sv = signature.getSignatureValue().validate(context);
        logger.info("Signature validation status: " + sv);

        // Check the validation status of each Reference
        Iterator<Reference> i = signature.getSignedInfo().getReferences().iterator();

        for (int j = 0; i.hasNext(); j++) {
          // signature was not valid so try to find out which refrence was invalid
          Reference ref = i.next();
          boolean refValid = ref.validate(context);
          String id = ref.getURI();
          logger.info("Reference (" + j + ") with URI [" + id + "] validation status: "
               + refValid);
        }
      }
      else {
        logger.info("Signature passed core validation!");
      }
```

3.1.2 Exemple: NamespaceContext Implementation

Ceci est l'application NamespaceContext qui connaît tous les espaces nominaux nécessaires de la réponse DTE. Le NamespaceContext est nécessaire afin que l'on puisse effectuer les interrogations XPath avec les préfixes correspondants. On assure ainsi le mapping entre les préfixes et les espaces nominaux.

```
public class NamespaceContextImpl implements NamespaceContext{

  public static final String NS_URI_WSSE = "http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd";
  public static final String PREFIX_WSSE = "wsse";
  public static final String NS_URI_SOAP_ENV = "http://schemas.xmlsoap.org/soap/envelope/";
  public static final String PREFIX_SOAP_ENV = "soap";
  public static final String NS_URI_XMLDSIG = "http://www.w3.org/2000/09/xmldsig#";
  public static final String PREFIX_XMLDSIG = "ds";
  public static final String NS_URI_EVV = "http://www.e-
dec.ch/xml/schema/edecReceiptResponse/v1";
  public static final String PREFIX_EVV = "evv";
  private Map<String, String> value = new HashMap<String, String>();

  public NamespaceContextImpl() {
    value.put(PREFIX_EVV, NS_URI_EVV);
    value.put(PREFIX_SOAP_ENV, NS_URI_SOAP_ENV);
    value.put(PREFIX_WSSE, NS_URI_WSSE);
    value.put(PREFIX_XMLDSIG, NS_URI_XMLDSIG);
  }

  public String getNamespaceURI(String prefix) {
    return value.get(prefix);
  }

  public String getPrefix(String uri) {
     throw new UnsupportedOperationException();
  }

  public Iterator<String> getPrefixes(String uri) {
    throw new UnsupportedOperationException();
  }
}
```

3.2 CRL – Certificate Revocation List (liste des certificats de sécurité révoqués)

Dans cet exemple de code, la CRL n'est pas encore vérifiée. On obtient une liste des certificats de sécurité révoqués en passant par la page Internet Admin PKI.

http://www.pki.admin.ch/crl.php

4 Sources

Spécification XML Signature Syntax and Processing (XMLDsig)
http://www.w3.org/TR/xmldsig-core/

Signature numérique XML API (JSR 105)
http://jcp.org/en/jsr/detail?id=105

Apache XML Security
http://santuario.apache.org/

IAIK XML Security Toolkit (XSECT)
http://jce.iaik.tugraz.at/sic/products/xml_security/xsect

Article: XML Signature with JSR-105 in Java SE 6
http://today.java.net/pub/a/today/2006/11/21/xml-signature-with-jsr-105.html?page=1

Article: Using JSR 105 with JDK 1.4 or 1.5
http://weblogs.java.net/blog/mullan/archive/2008/02/using_jsr_105_w_1.html

Présentation: XML Security and JSR 105-106
http://www.parleys.com/display/PARLEYS/XML+Security+and+JSR+105-106