Trustworthy Verification and Visualisation of Multiple XML-Signatures

Wolfgang Kubbilun[1], Sebastian Gajek[2], Michael Psarros[2], and Jörg Schwenk[2]

[1] MediaSec Technologies GmbH, Berliner Platz 6-8, 45127 Essen, Germany
wkubbilun@mediasec.de

[2] Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
{sebastian.gajek, michael.psarros, joerg.schwenk}@nds.rub.de

Abstract. The digital signature is one of the most important cryptographic primitives. It provides data integrity, message authentication and non-repudiation, which are required attributes in security critical services, such as electronic commerce, voting or health care. Whereas previous data formats for digital signatures concentrated on signing the entire document, the XML signature standard is feasible to secure complex workflows on a document with multiple signatures.
In a proof of concept implementation we demonstrate that verifying and trustworthily displaying of signed documents is realizable in standard Web browsers. The focus of our work are multisigned XML documents that introduce new requirements particularly in the field of presentation.

Keywords: Visualisation, WYSIWYS, XML, XML Signature, XPath, XSL Transformation, Web Browser.

1 Introduction

Electronic data exchange over TCP/IP networks made an overwhelming development in the past years and resulted in an increased need for security critical services. A major building block to secure documents are digital signatures. They have enormous practical relevance in fundamental services, such as electronic commerce, digital voting or health care. Signing and verifying of electronic documents is very crucial as it preserves the document‘s integrity and authenticity. For this reason, cryptographic algorithms (e.g., RSA, DSA, ECC-DSA) have been studied and are believed to be secure [1]. However, in practice this (single) assumption is insufficient to provide a sophisticated level of security. For instance, in legal proceedings one must also prove that the signed document has been presented correctly, i.e., one must show that the document and its content are clearly assigned to the signer (see e.g. [2]). As the typical user can only verify what she sees[1] the content of a digitally signed document must be visualised as well as information about the signer and the verification process (including the certificate chain).

[1] This is referred to as the *What You See is What you Sign (WYSIWYS) paradigm* (see section 2).

An additional aggravation is that in today‘s workflows several internal and external business instances process an electronic document. Generally, documents pass a hierarchical network of responsibilities (e.g., employees, supervisors) with different roles and access rights. To enhance the workflow (see section 4.1 for a further discussion) new constraints on digital documents have to be made: it is necessary to sign a document manifoldly. Multiple signing means to sign both the entire and certain parts of the document in unspecific order by multiple parties. Due to these requisitions of digital signing several new assumptions of the WYSISWS paradigm have to be made. Solving this paradigm in an user-convenient manner would dramatically enhance the use of digitally signed documents and broaden possible fields of application.

Our goal is to demonstrate the benefits of XML Signatures [3] and argue and elaborate that this technology is particularly suitable to design digital processes which require multisigned documents. More precisely, we employ XML and XSLT technologies [4] to fit the above named assumptions of the WYSWYS paradigm and mirror the results in a view, which can be presented, e.g., by standard web browsers. Finally, we show in a proof of concept that the Apache implementation [5] of the XML signature specification can be used for this approach.

The remainder sections are structured as follows: in section 2 we briefly introduce the presentation problem and discuss related works in section 3. In section 4 we propose an approach to solve the presentation problem based on XML technologies. In section 5 we prove the feasibility of our idea and provide a proof of concept implementation. Finally, we summarize our work and discuss future work in section 6.

2 The Presentation Problem

As mentioned in the introduction, cryptographically proven secure signature algorithms are not sufficient for providing an over-all security in practice; the presentation of the signed document is also decisive. A general problem results from the fact that electronic, respectively, electronically signed documents are processed by machines and displayed in an user-compatible manner, i.e., machines interpret data to the user’s convenience and adequately present it [6]. However, this presentation can be incomplete, incorrect or ambiguous, which yields to a mis-interpretation of significant information including a distorted view of the document‘s integrity and authenticity. This is called the presentation problem [7,8,9,6]. For a trustworthily presentation of digital signatures, further assumptions[2] have to be fulfilled, which we briefly summarise:

The presentation must be *definite*. In particular, the presentation of signed data must be unambiguous. A verifying person must always be aware of what has been signed. In practice, she is restricted by several factors. Depending on, for example, the displaying device or layout she might be interfered to see the

[2] We concentrate on basic problems, as a complete discussing would go beyond the paper‘s scope.

proper document. A frequently mentioned example is the use of white font on white background, which hides the actual content. Moreover, active content is (in this context) crucial. On the one hand it provides a level of dynamics, which is required today for a more user-friendly presentation. On the other hand this level of dynamics might inherent the capability to disguise the actual content's presentation [10].

In addition, the presentation must be *transparently.* This issue is particularly essential in legal disputes proving non-repudiation: surveyors must trace that the document led to this presentation. In cases of a dispute, the presentation must be reproducible. This is only feasible, if the document format is known, i.e., if each byte and its function is disclosed. For this reason, a definite separation of content and presentation enables a more convenient solution to a higher level of transparency.

Perhaps the most important aspect, specifically when considering an entire system, is that the document must be displayed *securely.* So far, we considered the semantic and syntactic requirements of a document format. We did neither consider the application nor the system, which are threatened by certain attacks (e.g., Trojan horses [7]). An electronic document can only be trustworthily displayed when the intrinsic system is not compromised and sound. In other words, the system and its application are trusted. This is an ultimate prerequisite, or else a digital signature is not tamper-proof. Note that the argumentation is invertible: a trusted system, respectively, an application on its own is insufficient to present an electronic document trustworthily. In addition a suitable document format is needed, which enables the document's content to be displayed unambiguously (as discussed before).

In the case of multiple signatures, these requirements become more complex. As a trusted system is not part of this paper, we presuppose an uncompromised and sound system for the present work and concentrate on definitiveness and transparency.

3 Previous Works

In the past, several commercial and open document formats/standards have been proposed (see e.g. [11,12,13,14]) which are able to present electronically signed documents.

A basic approach is to transform the content into an image (see e.g. [15]). The user sees a static effigy of the document. The benefit of this approach is that a static document does not contain any active content, i.e., it circumvents the possibility of altering the content. This approach is from todays point of view impracticable as static documents are hardly editable, hardly processible and unqualified for multi-party business models. Another basic approach is encoded text as used in common signature standards (e.g. S/MIME [12], CMS [13]). However, the encoded text based standards have shortcomings. They define how to embed the signature—even multiple signatures (cf. [14])—into the document format. In contrast to XML, they do not define how to present the content;

the presentation depends on its application and, hence, is interpreted product-specifically.

The most tangential work was proposed by Scheibelhofer [8]. First of all, he deployed the benefits of XML technologies in the context of signing and verifying electronic documents. He developed an Internet terminal [16] aiming at signing, validating and trustworthily displaying electronic documents. Mainly, his approach is construed for single-signed documents. Our idea goes beyond. Although we use similar techniques, we also take into account the usage of multiple XML signatures. Furthermore, we do not provide an architecture for a signing terminal, instead we demonstrate that validating and visualising XML-signed documents is feasible in standard Web browsers.

4 Verification and Visualisation of Signed XML Documents

4.1 The Need for Multiple Signatures

Many business processes require multiple signatures. Usually the responsible persons do not sign at the same time and the same place. For example, the business process of creating an invoice by a fictive software company might require four different signatures: the confirmation of a person that certain goods have been packaged and shipped, the responsible sales person signing for special agreements that have been made with the customer, another signature by a controlling instance (the company's invoice department) and, finally, the signature of the general manager. Nevertheless, the different process participants are not willing—and partly not able—to sign the document in its completeness due to their restricted areas of responsibility. In this sense the person responsible for packaging for example will not be willing to sign special sales conditions. Therefore, if such a workflow is mapped to the digital world, the according responsibilities have to be mapped too; otherwise the new solution will not have the needed acceptance by all participants.

XML signatures provide two features which make them especially attractive for application scenarios that require complex signing processes: the use of XPath [17] expressions and the support for multiple signatures. The XML signature standard [3], published in february 2002 by the World Wide Web consortium as a W3C recommendation, defines an XML based format for digitally signed data. The data to be signed is referenced through Uniform Resource Identifiers (*URI*s) and may be XML as well as arbitrary digital content. The use of URIs enables the signing of external data sources like file or network resources. The XML signature is calculated over a list of XML references of which every reference contains the URI, the used hash algorithm and the hash value.

Additionally, the XML standard lists five different transformations—including XPath—which can be applied to the XML reference to be signed.[3] The

[3] Note that not all of these transformations are defined to be required. The implementation of XPath for example is *recommended* but not required by [3].

XML reference is first transformed before the according hash value is calculated. Hence, the transform or list of transforms respectively, act as a pre-processing step within the XML signature creation process. This fact has a great impact on the visualisation of the XML signature: the data secured by the signature might differ from the referenced source. This aspect is also addressed explicitely by [3] (see [3] chapters 8.1.2 and 8.1.3).

The primary purpose of XPath is to identify particular parts of an XML document (or more precisely: subsets of the XML document's node set). Furthermore, XPath also provides some basic operations for the manipulation of strings, numbers and booleans. As a conclusion, XPath expressions are sufficient to identify definite parts of a document that have to be signed.

An XML document may contain more than one signature. In particular, if it comprises some signatures securing parts of the document by the use of XPath, another signature covering the entire document's content may additionally be inserted. Every signature is represented by a `<Signature>` tag, which hosts any relevant data, such as key material, references and their according digests. Therefore each XML signature can be validated independently from the other ones.

The named properties of the XML signature are powerful features and, if properly used, enable an appropriate design of business processes in the digital domain. Nevertheless, they also result in new requirements regarding the validation and visualisation process of a signed XML document[4]: if the content of a PKCS#7 container comprising multiple signatures is manipulated all signatures will become invalid. In contrast to that a signed XML document may contain both valid and invalid signatures. The use of URIs and transformations in the XML references can additionally complicate the presentation problem, especially regarding the aspects of definitiveness and transparency. In the following we present an approach to overcome these problems. One important design goal thereby is the use of standard Web browsers for the verification and visualisation of the signed XML documents.

4.2 Visualisation Based on XSL Transformations

We propose to utilize the XSLT technology in order to present the signatures of an XML document to a human viewer. XSL transformations defined by [4] operate on an XML source tree and generate a result tree (possibly but not necessarily in XML format). The transformation is achieved by patterns defined by the XSLT, which are applied to the input. A popular application of XSLT is the transformation of XML input into HTML or XHTML output. We also make use of XSLT to generate a compact view of the XML signatures' signed data that can be displayed by a standard HTML browser. Moreover, we use XSLT to mark the signed data parts of the XML signatures for the visualisation, which will be explained below.

We describe our approach in a more detail before we discuss its advantages. Fig. 1 depicts the workflow representing our main idea: First the signed XML

[4] In the following a *signed XML document* means an XML document containing signatures according to the XML signature standard [3].

input document is validated against the XML schema defined by [3]. This step assures the syntactic correctness of the input. After that each XML signature of the input document is evaluated. The results comprising the signature's status and information about the signer are delivered to the XSLT facility[5]. Then the XPath expressions of the signatures, which define their signed data (in our application environment[6]) are extracted from the according `<Reference>` tags and passed to the XSLT component as well.

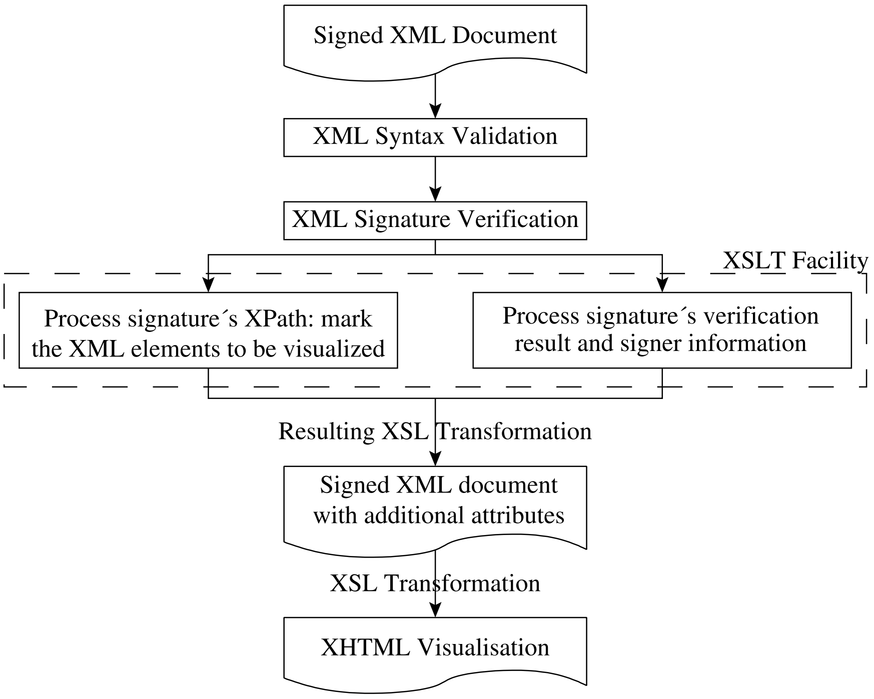

Signed XML Document
XML Syntax Validation
XML Signature Verification
XSLT Facility
Process signature´s XPath: mark the XML elements to be visualized
Process signature´s verification result and signer information
Resulting XSL Transformation
Signed XML document with additional attributes
XSL Transformation
XHTML Visualisation

Fig. 1. Visualisation workflow for a signed XML document

The XSLT facility takes the original signed XML document as a basis for the resulting view of the document. It processes the XPath expressions of each signature to determine its signed data to be visualised. During this step every XML tag that belongs to the signed data part is provided with an additional attribute called `signatureID`. The value of this attribute is set to an unique id (e.g., the name of the signer), so that we are later able to distinguish the signed data parts of different signatures in the resulting XML document. After

[5] The XSLT facility is the component of our system, which is able to process XSL transformations.

[6] Note that in the given context we have documents in mind, which only use XPath as transformation—as already indicated in Section 4.1. Nevertheless, the outlined approach can be extended to cover additional transformations defined by [3].

processing all signatures this version of the signed XML document is stored as an intermediary result.

The intermediary result represents a machine readable version of the visualisation that we display to the human viewer. The values of the `signatureID` attributes act as a kind of highlighting in XML format. In the next step this highlighting must be transferred to a human readable view. Again we use an XSL transformation to produce an XHTML version of the original signed XML document. The highlighting given by the `signatureID` attributes is visualised and connected to the signer's identity. An example of such a visible highlighting is a box that is drawn around the signed data. Based on the status of the signature that the XSLT facility gets as input the highlighted parts of the XML document can be displayed in an adequate manner, such as a green box in case of a valid signature and a red box otherwise.

Our approach fits the requirement to present the validity of each XML signature contained in the input file individually to the human viewer. Moreover, the signed data of a given signature is visualised within its textual context. Thereby, the model of authorisation that comes along with a multisigned document is mirrored by our view. Another advantage is that any standard Web browser can display the XHTML result. This aspect leads to the idea to construct a Web based service, which offers verification services for signed and multisigned XML documents, i.e., to offer a centralised XML verification. Such a service may be offered either by a PKI operating entity in a local or by a trusted third party (e.g., a trust center) in a global application environment.

The use of the XSLT technology to prepare the visualisation for the human viewer facilitates particularly the support of different display devices. Based on the machine readable version of the visualisation different XSL transformations can be supplied to present the visualisation results also on devices with limited display capabilities (e.g., mobile phones). The format of the presentation results is thereby not restricted to XML or XHTML.

5 Proof of Concept Implementation

We use the Apache Java implementation [5] of the XML signature standard and the Apache XSLT processor Xalan [18] for our proof of concept realization. First we implement a Java module that takes a signed XML document as input, parses it, finds all contained XML signatures and finally does the validation of the signatures. This module makes intensive use of the named Apache library to do the verification.

After that the module extracts the XPath expressions that have been used during the creation of the XML signatures. In order to do so it loops through all references of a signature's `<SignedInfo>` element and identifies transformations of type XPATH. To export the xpath transformations out of the XML document the Java module creates an output file in form of an XSL transformation. The XPath expressions of the signed XML document are then written to the XSL output file. Thereby, the according XSL patterns are constructed to attach an unique signer id to each elements belonging to the signed data part of the signature. In detail we write a `<template>` element to the output file whereas we set the `match` attribute to the exported XPath while the content of the element is set to the signer id. To simplify the implementation we use the id attribute of the signature's `<SignedInfo>` tag even if this attribute is declared to be optional by the XML signature standard [3]. Furthermore, we output the XPath expressions of a given signature only if the validation process of this signature evaluates to be valid. In other words, we only highlight the valid signatures of an input document.

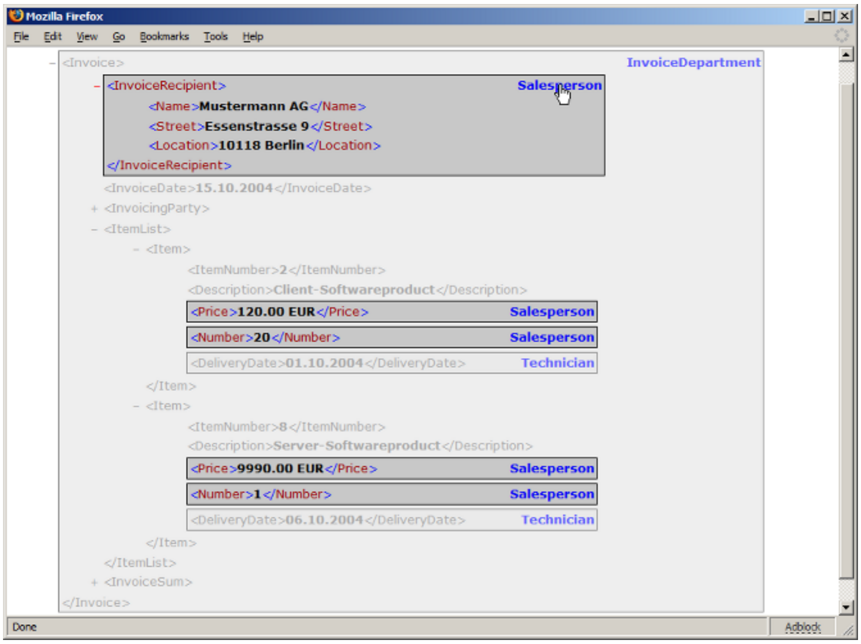
After the processing of the Java module has been finished the resulting XSL transformation is passed to the Xalan library (building the XSLT facility in our implementation). The transformation is then triggered by the Java module. The resulting XML document represents the machine readable version of the visualisation which we mentioned in Section 4.2. Regarding the representation for the human viewer we decided to develop an XSL transformation that transfers the content data of the XML document into an XHTML version. The signed data part of each signature is thereby surrounded by a box. Additionally, the text content of a border is displayed in dark colours if the according signer id is clicked whereas all other text of the document is faded out. Nevertheless, this is just a matter of design. Other forms of the visualisation are imaginable – dependant on the capabilities of the targeted display device – provided that the assignment of the signed parts to the signer is obvious.

As a sample business process requiring multiple signatures we created a fictive invoice document in XML format (see Fig. 2). We attached three different XML signatures to the document:

– the signer (with the id *Technician*) confirms that the invoiced software packages have actually been delivered,
– the signer *Salesperson* electronically signs the agreed prices as well as the invoice recipient's contact details and
– the signature *InvoiceDepartment* finally covers the entire document and confirms that the invoice document has correctly passed the prescribed workflow. In addition it guarantees the integrity of the document as a whole.

We applied our proof of concept implementation to the fictive invoice document and generated the according XHTML document. This document mirrors the different roles and the assigned authorisations of the signers: the technician, for instance, is not able to confirm the correctness of the pricing information, so she does not sign for it. It is therefore essential that the visualisation of her signature unambiguously presents to the viewer exactly those parts of the document that she actually signed. Our prototype implementation fits this requirement.

Fig. 2 shows the result when the signer id of the sales responsible is clicked: the numbers, prices and the invoice recipient are displayed in darker colours meaning that the sales responsible signed exactly these data. The delivery date of the invoiced products, for example, which is faded out and surrounded by another box is not part of her signature. We tested the output document with the



```
- <Invoice>                                                        InvoiceDepartment
    - <InvoiceRecipient>                                 Salesperson
          <Name>Mustermann AG</Name>
          <Street>Essenstrasse 9</Street>
          <Location>10118 Berlin</Location>
      </InvoiceRecipient>
      <InvoiceDate>15.10.2004</InvoiceDate>
    + <InvoicingParty>
    - <ItemList>
        - <Item>
              <ItemNumber>2</ItemNumber>
              <Description>Client-Softwareproduct</Description>
              <Price>120.00 EUR</Price>                  Salesperson
              <Number>20</Number>                        Salesperson
              <DeliveryDate>01.10.2004</DeliveryDate>    Technician
          </Item>
        - <Item>
              <ItemNumber>8</ItemNumber>
              <Description>Server-Softwareproduct</Description>
              <Price>9990.00 EUR</Price>                 Salesperson
              <Number>1</Number>                         Salesperson
              <DeliveryDate>06.10.2004</DeliveryDate>    Technician
          </Item>
      </ItemList>
    + <InvoiceSum>
  </Invoice>
```

Fig. 2. Visualisation of signed XML invoice document

Mozilla Firefox (version 1.0) and with the Microsoft Internet Explorer (version 6) browsers.

6 Conclusion and Future Work

Our investigations of the XML signature standard and especially the implementation that we described in section 5 demonstrate the suitability of the XML signature for multisigned XML documents. XML signatures allow the design of digital workflows considering the different roles and according authorisations of the signers. The XSL transformations turned out to be an appropriate technology to realize a browser based visualisation system for signed XML documents. Moreover the Apache Software Foundation provides an usable implementation of the XML signature standard with [5], as our visualisation prototype shows.

Future research includes the investigation of more application examples that need multiple signatures and to define adequate visualization styles for them. Another area is to clearly define enhancements for current browsers (e.g., inclusion of XML signature verification) such that the display of signed documents on the WWW will be possible by simply providing a signed XML document and a (signed) XSLT transform.

The authors want to acknowledge the contribution of Lars Ewers regarding the proof of concept implementation of our approach.

References

1. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton, Florida (1996)
2. European Parliament and Council: Directive 1999/93/ec of the european parliament and of the council of 13 december 1999 on a community framework for electronic signatures. Official Journal of the European Communities (2000)
3. The W3C: XML-Signature Syntax and Processing, W3C Recommendation. (2002) http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/.
4. The W3C: XSL Transformations (XSLT), W3C Recommendation, Version 1.0. (1999) http://www.w3.org/TR/1999/REC-xslt-19991116.
5. The Apache Software Foundation: Apache XML Security API, Version 1.1.0. (2004) http://xml.apache.org/security.
6. Pordesch, U.: Die elektronische Form und das Präsentationsproblem. Nomos Verlagsgesellschaft (2002)
7. Weber, A.: See what you sign: Secure Implementations of Digital Signatures. In: International Conference on Intelligence and Services in Networks. (1998)
8. Scheibelhofer, K.: What You See Is What You Sign - Trustworthy Display of XML Documents for Signing and Verification. In: Communications and Multimedia Security. (2001)
9. Spalka, A., Cremers, A., Langweg, H.: The fairy tale of 'what you see is what you sign' - Trojan Horse Attacks on Software for Digital Signature. In: IFIP WG 9.6/11.7 Working Conference. (2001)
10. Kain, K., Smith, S., Asokan, R.: Digital Signatures and Electronic Documents: A Cautionary Tale. In: Communications and Multimedia Security. (2002)
11. Callas, J., Donnerhacke, L., Finney, H., Thayer, R.: OpenPGP Message Format. Network Working Group. (1998) Request for Comment 2440.
12. Hoffman, P.: Enhanced Security Services for S/MIME. Network Working Group. (1999) Request for Comment 2634.
13. Housley, R.: Cryptographic Message Syntax. Network Working Group. (1999) Request for Comments 2630.
14. Kaliski, B.: PKCS #7: Cryptographic Message Syntax Version 1.5. Network Working Group. (1998) Request for Comment 2315.
15. Utimaco AG: WYSIWYS - What you see is what you sign. (2003) http://www.utimaco.com/eng/content_pdf/wysiwys.pdf.
16. Scheibelhofer, K.: Signing XML Documents and the Concept of "What You See Is What You Sign". Institute for Applied Information Processing and Communications, Graz University of Technology. (2001)
17. The W3C: XML Path Language (XPath), W3C Recommendation, Version 1.0. (1999) http://www.w3.org/TR/1999/REC-xpath-19991116.
18. The Apache Software Foundation: Apache Xalan-Java, Version 2.6.0. (2004) http://xml.apache.org/xalan-j/.