What You See Is What You Sign

Trustworthy Display of XML Documents for Signing and Verification

Karl Scheibelhofer
Institute for Applied Information Processing and Communications
Inffeldgasse 16a, A-8010 Graz
Graz University of Technology
Email: Karl.Scheibelhofer@iaik.at

Key words: trusted viewing, trustworthy display, electronic signature, XML documents, secure transformation, signed XML schemas, signed stylesheets, multiple platforms, easy extensibility

Abstract: This document shows a solution to display documents in a trustworthy manner. The application focuses on systems that are used for signing and verification, because trustworthy display is of particular interest in the area of electronic signatures. Moreover, this paper focuses on XML [1] as format for encoding documents. The paper shows how a display module can apply advanced filtering techniques. Such advanced filtering techniques can ensure a maximum of security for displaying documents. A system built on this approach could assist the user much more than traditional systems can. Hence, the user needs to take less care in everyday business, when executing sensitive tasks like signing electronic documents. Furthermore, such a system would be easier to manage, because most functionality is generic. Thus, extending the system to handle new types of documents does not require installation of new code.

1. INTRODUCTION

On December 13th, 1999 the European Parliament and the Council of the European Union established the Directive on Electronic Signatures [5]. Several countries of the European Union established electronic signature

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35413-2_36

R. Steinmetz et al. (eds.), *Communications and Multimedia Security Issues of the New Century*

laws to implement the directive [7]. Having a legal basis, electronic signatures are lacking practical implementations. Building a system that can create advanced electronic signatures is a complex task. Such a system must meet several requirements to enable users to create signatures in a secure way. One of the most crucial requirements is that the system must display all data that the user wants to sign. Moreover, the system must display the data in a manner that the user can perceive and understand the content. Achieving this can be very expendable. Nowadays, document formats in use have the property that they contain the data and presentation information mixed up in a single document. Examples for such formats are PDF, Word or HTML. In addition, these formats allow content that can be risky to sign. Just consider macros that present different information depending on time or any other external state. To sign such documents in a trustworthy way, the system needs to filter any questionable content before presenting the document to the user and before creating the signature. But active content is not the only threat. Any document mixing up content and presentation can easily include white text on white background. Thus, a user might sign content that he has never seen. Filtering such documents for problematic content is not only complex to handle, it is also not very suitable for automated processing.

The approach presented here uses a completely different way. Using a document format that supports strict separation of content and presentation has several advantages over solutions working with other document formats. The document format used here is XML. An XML document holds just the raw data but no information how this data should be presented. There are separate documents that describe how an XML document of a certain class should be presented. Such presentation documents, often called stylesheets, transform the XML document into a format that the destination system can display. Normally, a stylesheet produces correct output just for one destination device. Thus, we need different stylesheets to display the same XML document on different platforms. This gives the advantage that we can use a presentation appropriate for the current display device. Nevertheless, writing such stylesheets, we must ensure that the documents they produce convey the same information. Information in this context refers to the semantic of the content rather that to information in the sense of information theory.

The system presented in this document is a solution for signing XML documents in a trustworthy manner. This solution can be administrated centrally. Moreover, it can be used on a various different platforms. While ensuring correct display of documents on all platforms, it remains general and easily extendable. It is easy to extend the system to handle additional classes of XML documents. There is no need to recompile any code, when extending the system to handle new document types. This can be done at runtime seamlessly.

2. TRUSTWORTHY DISPLAY OF XML DOCUMENTS

The system presented focuses on document formats that separate content and presentation. For simplicity, we use only XML here. However, the concept should be easily adaptable to any format that separates content and presentation.

It is a fact that different devices have different capabilities determining what they can display and how they display things. Supplying presentation information in a document implies that the author of the document knows the device the document will be displayed on. Moreover, he also fixes his document to this one type of device. For instance, he decides to give one heading red colour. If any user tries to view this document on a monochrome display, what will the browser do? Will it display the heading in black, in grey, or not at all?

A simple way to cope with this problem is to separate content and presentation. In practice that means splitting the document into two documents. The first part only holds the content in a structured manner, and the second holds presentation information. The presentation information in the second document just describes how the content should be presented. This could be information like: display all first level headings using the Arial font with a size of 16 points and in bold face. A document that only bears presentation information is often called a stylesheet. A stylesheet can be viewed as a transformation from one specific format to another specific format.

But one must keep in mind one important fact: in general, a stylesheet only works with one type of document and it only produces useful output on one device. For example, a system that deals with examination certificates in electronic form needs a separate stylesheet for every display device an examination certificate document should be displayed on. If the system also wants to deal with birth certificates, it needs another set of stylesheets for all devices in use. In contrast to the additional effort, this brings the advantage that the document can be displayed on the concerned device in an optimal way. A stylesheet can exploit the strengths and avoid the weaknesses of the display device.

Another advantage of strictly separating content and presentation is that a system only needs to handle pure content inside the workflow system. Structured content is easier to handle for computers, if it is not mixed up with presentation information. Presentation information is useless for software; its only purpose is presenting a document to humans. Presentation information is only necessary, where data needs to be presented to human eyes, ears or any other sense.

For signature terminals, it is necessary to have a system that displays documents accurately. A signature expresses a type of commitment to some content. But if I cannot see the content, because the system does not display it correctly, I cannot make a commitment. Moreover, if the system displays the contents of a document not accurately or not completely, I might commit to contents that I have never seen. Using stylesheets gives us a powerful tool to display documents correctly. For the system, a stylesheet behaves like a transformation that the system loads at runtime on demand.

Some reader might ask now, if the system cannot display the document as is. This would mean displaying the document's encoding as plain text or, even worse, as hex dump. Normally displaying a document this way is very inconvenient, even for rather simple formats like HTML. Critics might reply that this would avoid any source of misinterpretation. But that is not true. If you view documents or data in general, it is always only one view. Information is an abstract thing. Hence, information has no unique presentation in general. Even if the program tries to display a document as bit-stream, it must define how to present a set bit (a one) and an unset bit (a zero). Of course, in practice, usually a one ('1') represents a set bit and a zero ('0') represents an unset bit, but nothing prevents me from defining it differently. On the other hand, it is not reasonable for most users to present them the encoding of a document in plain text or even in binary format. Consequently, the only real requirement is: Different presentations of the same document must convey the same meaning to the user.

So we can sum up: It is always necessary to get an appropriate representation for a document. The representation must present the complete content of the document to the user, and it must do that in an unambiguous and comprehensible manner. To interpret the content of a document correctly it is necessary to know how the content is encoded. There are numberless many different formats around that are used to encode information. We will focus on XML as format for documents that we want to display. Using XML, it is easy to separate content and presentation of documents.

Signing XML documents, we need additional information that tells us how to display a certain XML documents. Stylesheets carry such information. For simplicity, they are XML documents themselves. There is a standard for XML stylesheets that is called XSL [4]. Normally, a stylesheet does apply only to one type of XML documents; for instance, it can apply to XML documents that are valid according to an XML Schema [2]. A stylesheet does not actually tell a program how to display a document; rather it is a kind of transformation. It defines how to transform an XML document to a document that contains formatting information. The destination format of such a transformation can be any format. For simplicity, formats like HTML or RTF are used, because it is relatively easy to get a viewer for these

formats. Using stylesheets for the transformation, it is quite easy to avoid features of the destination format that may cause trouble. A developer can simply avoid using these features the stylesheets.

2.1 Filtering the Document

Besides considering the process of transformation just as a mapping from one format to another, we can look at it as a process doing some kind of filtering. Because a transformation normally only works for one type of documents, we need to ensure that the documents passed to the transformation are of this specific type. This document type filter just mentioned comes before the actual transformation of the document. Depending on the application and type of transformation used, it might be necessary to insert some more filters before the transformation. For instance, if we use stylesheets in the transformation, we need to assure that there is an appropriate stylesheet available. Moreover, the application could display a warning to the user, if he is going to view a document that requires special education to understand and the user does not have it. After applying the transformation, we get a new document as a result. Likewise, the application might filter the transformation result in an analogous way (see Figure 1). For example, a user capability filter at this stage could check, if the text in the result document is written in a language that the user understands. Having XML as document format in mind, the following sections explain what each filter does. The last paragraph suggests some ides for additional filters.

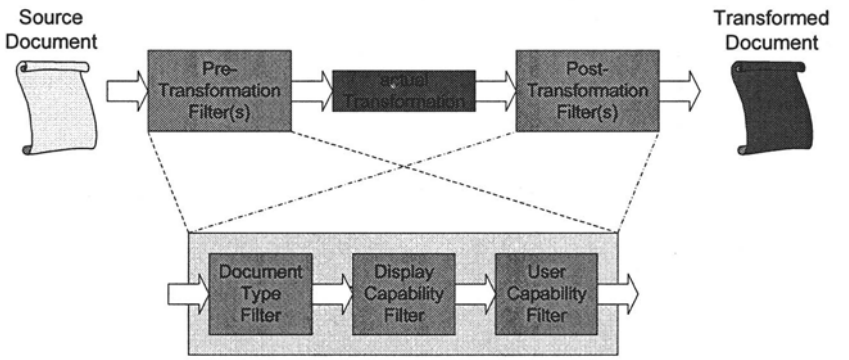

Figure 1. Filters before and after signing

2.1.1 The Document Type Filter

This filter determines if the document is of a type that the software can handle. For the case that the data is XML data, this would cause the software to look for the specified DTD or XML Schema of the document and accept it, if it is trustworthy; this is, if it is signed by a trusted entity. If there are further requirements to a document to make it a valid document of a certain type, this must also be checked in this stage. For instance, the application could look up the values of certain fields in a database to check if they are valid. Applied after transformation, this filter might check the result to be an element of a defined set. Taking a subset of XHTML as destination language, this filter could easily validate the document according to this subset.

2.1.2 The Display Capability Filter

The display filter checks what the display capabilities are and searches for trusted XSL stylesheets; this means, the stylesheet is certified to transform the given type of document in a trustworthy manner to an appropriate destination format for the display device. This is also the right place to check the documents for other properties that might influence, if they can be displayed correctly. For instance, the program will have to check, if the used character sets are available on this device.

2.1.3 The User Capability Filter

Filtering data with respect to the hardware and software capabilities is not enough, because this can only assure that the data can be displayed correctly. The user must be able to understand the document he wants to sign. Roughly, the same considerations and restrictions as with the device might apply. The user's capabilities can be separated into two categories. First, there are physical capabilities that enable or prevent the user to see and read the document. Second, there are mental capabilities that enable or prevent him to understand the meaning of the document. Below, there is a list of the most frequent capabilities that an application needs to consider.

- Physical capabilities
  - Handicaps
  - Blindness
  - Deafness
  - Colour Blindness
- Mental capabilities
  - Languages the user can read and understand
  - Education

There are much more capabilities and disabilities that an application can take into account. It will also heavily depend on the document and its application how detailed all these capabilities need to be.

2.1.4 Additional Filters

There are additional filters imaginable. Here we list just a few ideas for such filters. An additional filter could be a semantic analysis filter that is capable to check if there are any inconsistencies in the document. This would mean that the filter checks, if a contract does not state facts in one chapter and states completely contradictorily facts in a later chapter. Semantic analysis of documents is a very complex task and thus it might not be feasible in most environments.

An easier filter could be a simple spell and grammar checker ensuring syntactic correctness. This filter could also check if the document is really written in the language that is claimed in the meta information of the document. For instance, if the document meta information claims that the document is in English and the spell and grammar checker finds out that it is actually written in German it could inform the user. Such a filter can also check if all abbreviations used in this document are defined before their first use. In case of a dispute, such details could be important. Therefore, it is desirable that the system supports the user as much as possible.

2.2 The Display Device

This chapter explains the display device in more detail. In this context the display device is considered as the combination of hardware and software that is necessary to display data. Data the device can display would normally be a document of some specific format; for example, the format could be a subset of XHTML.

The data that passed all filters is now presented on the display device using the information gained through the previous steps of filtering and transformation. The display device with the viewer software must be able to display all contents that will be signed, without any exception. Otherwise, it must reject the document or at least warn the user. First, the capabilities of the display have to be available to the software. That is necessary to decide, if certain data can be displayed on this device.

The appropriate stylesheet describes how documents of a given type can be transformed to a format that this device can display. For example, if the document is an XML document, the stylesheet would be a XSL document. On a mobile device with a small display, the system would take this stylesheet and transform the XML document to a simple text document with

some simple formatting information. Hence, the result of the transformation could be a plain text document with formatting tags that are taken from a subset of HTML or RTF. The result format always depends on the display device, because it needs to be a format that the device can display. Even though, the result format will normally be a high level format and not a very low level one like a fully rendered bitmap. The display device will still do the rendering. Stylesheets will have to be authentic, thus signed by a trusted entity. By singing a stylesheet, this entity guarantees that this stylesheet is appropriate for transforming documents of a specific type to a specific format for a specific device. This means, there is one stylesheet for every pair of document type and device (see also Figure 2).

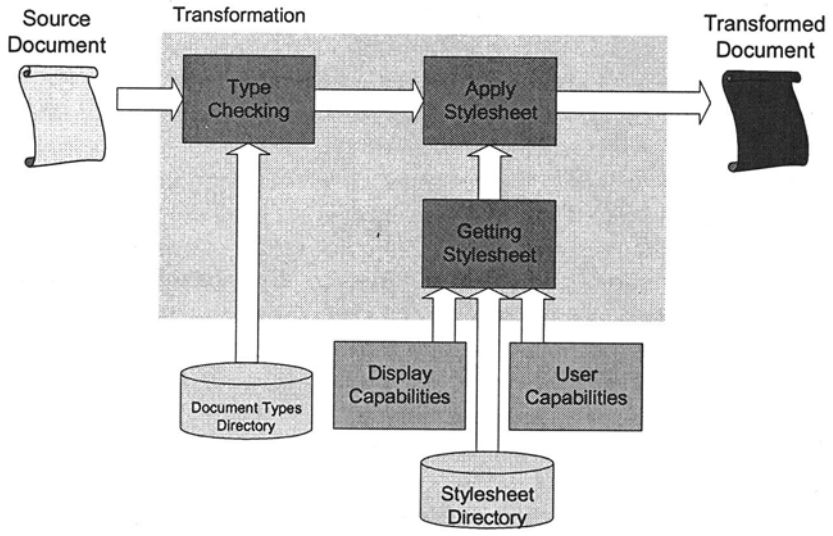

Figure 2. The transformation for XML documents realized with stylesheets

In some cases, it might be desirable to force the user to display all data and prevent him from skipping parts. Thus, a user cannot argue that he has not seen all parts of the document. This point is likely to be handled by means of software. It should be ensured that the user really read all parts of the document and did not skip some parts due to hurry or laziness. Perhaps this requirement can be relaxed so that the user must be able to read all parts, but he is not required to do so. Simply consider a boss signing documents presented by his assistant, he does not want to read the same documents again and again every time he sings one.

2.3 Security Considerations

Since this system is intended to be used in terminals generating secure digital signatures, it must meet certain security requirements. We roughly distinguish between the components: hardware, operating system and software.

2.3.1 Hardware

Of course, the hardware used must offer protection against certain attacks. But the hardware is outside the scope of this document. Most parts of computation relevant for generating the keys and generating the digital signature are done on a smart card or on some other security token. But that does not mean that the other hardware involved is not required to be secure. Of course, it has to be. Simply consider the case in that the hardware sends some completely different data to the smart card, than the one you looked at. For example, the hardware could provide an exhaustive self-test that ensures that the hardware was not compromised.

2.3.2 Operating System

The operating system supplies the basic functionality and has access to all system resources. Thus, it is critical from the security point of view. Note that the data that is subject to singing passes through parts of the operating system, when the application sends it to the smart card. In addition, we have to rely upon the operating system's display functions; for example, if we tell the operating system to display a line of text, we have to rely upon the correct functioning. It is desirable that the operating system has also some self-test mechanism. This could be achieved with signing the different parts of the operating system and verifying the signatures during system start-up. Additionally, the operating system should provide a feature to verify a signed application before it executes it. It should at least prevent the installation of application code that has no valid or no trusted signature. However, this part is also not covered in more detail by this document and is left up to the developers of the hardware and operating system.

2.3.3 Software

All application code must be authentic and trusted. An easy way is to sign the application code. Consequently, the operating system must verify the signature of an application before it executes it. In some situations, it is enough to verify the signature during installation of the software. In this

case, the system is required to absolutely prevent the installation of any unsigned code and modification of installed and verified code. Not only the operating system must verify the authenticity of data supplied by external source. The application must also verify any data and document that it retrieves from external sources. Such external sources can be network drives, web servers or floppy disks. If the application uses an XML parser it must ensure that all additional data the parser retrieves during operation are authentic and trusted. To be more concrete, the application must ensure that all grammars (DTDs or XML Schemas in context of XML) and stylesheets used are trusted. To achieve this, all these documents can be signed. Very often, it will be necessary that not only the document itself is authentic but also the meta data associated with it. For instance, a signed stylesheet must carry additional attributes that tell the application to what type of documents and to what devices the stylesheet applies. Furthermore, the stylesheet may explicitly state the language of the output it produces. Implementing the transformation using signed stylesheets has a further important advantage. A trusted authority can design a stylesheet. By signing the stylesheet with certain attributes, the authority certifies that this stylesheet produces accurate output for a specific class of documents on a certain class of devices. Consider a sales contract. First, a ministry would define a grammar for a sales contract. For XML, the grammar would result in a DTD or XML schema document. Having the grammar, the ministry could issue a stylesheet that is certified to produce an accurate PDF document from an XML sales contract. Thus, anyone could create a sales contract according to the defined grammar. But nobody can influence the presentation of a sales contract, even not the creator of the sales contract himself. This reduces the options for fraudulent sales contracts.

3. CONCLUSION

We can distinguish two different approaches. One using formats mixing content and presentation as format to encode and sign documents. PDF is such a format. The other approach uses a format that separates content and presentation. XML strictly separates content and presentation. Separating content and presentation has several advantages over mixed formats. Documents containing just the content in a structured form are easier to handle for computer programs. Moreover, having separate presentation information enables us to provide different presentation methods for different devices. Thus, the same document can be displayed (or printed) on a high-resolution graphic device, a monochrome test display and a laser printer. To do this we provide three different presentation documents that

describe how to present the document on the destination device. In the context of XML, such presentation documents are called stylesheets. These stylesheets can be signed, and thus they certify that they produce a correct output for a certain type of documents on a certain device. Moreover, the presentation is under exclusive control of the stylesheet issuer, assuming that the whole system is secure. Over all, the presented document offers a flexible and scalable solution for big heterogeneous environments. The solution applies to various devices reaching from powerful desktop computers to mobile devices with limited resources.

4. REFERENCES

[1] The W3C, XML 1.0, 10. February 1998, available online at http://www.w3.org/XML/

[2] The W3C, XML Schemas, Candidate Recommendation, 24.October 2000, available online at http://www.w3.org/XML/Schema

[3] The W3C and the IETF, XML Signature, Candidate Recommendation, 31.October 2000, available online at http://www.w3.org/Signature/

[4] The W3C, Extensible Stylesheet Language (XSL), Candidate Recommendation, 21.November 2000, available online at http://www.w3.org/Style/XSL/

[5] The European Parliament and the Council, "European Directive on Electronic Signature", Brussels, December 1999, available online at http://europa.eu.int/ISPO/ecommerce/legal/digital.html

[6] Electronic Telecommunications Standards Institute, "Electronic Signature Formats", France, 2000, available online at http://www.etsi.org

[7] The National Council of Austria, "The Austrian Signature Law", Vienna, August 1999, available online at http://www.a-sit.at/

[8] The Chancellor of Austria, "The Directive on Signatures", Vienna, February 2000, available online at http://www.a-sit.at/