What You See is Not Always What You Sign*

Audun Jøsang

Distributed Systems Technology Centre †
Level 12, S-Block, Queensland University of Technology, GPO Box 2434, Brisbane Qld 4001, Australia
tel:+61-7-3864 5120, fax:+61-7-3864 1282
email: ajosang@dstc.edu.au

Dean Povey and **Anthony Ho**

Wedgetail Communications
Level 14, 388 Queen Street, Brisbane Qld 4001, Australia
tel:+61-7-3023 5100, fax:+61-7-3023 5199
email: dean.povey@wedgetail.com, anthony.ho@wedgetail.com

Abstract

A fundamental aspect of computer systems is that displaying and signing a digital document are separate and unlinked processes. In addition, the same digital document can be displayed differently on different systems. As a consequence it is difficult to determine what exactly has been signed, both from the signer's and the verifier's point of view. This paper discusses how confusion about the meaning of digitally signed documents can occur, and proposes some mitigation strategies.

Keywords: Digital signature, XML, HTML, ASN.1, web, e-commerce, security

1 Introduction

The concept of "digital signature", first publicly described by Diffie and Hellman (1976) [4] in their classic paper "New directions in Cryptography", suggests that it is a computer-based equivalent of physical written signatures. Although there are similarities between handwritten and digital signatures there are also fundamental differences. The main similarity is that both types of signatures can provide evidence of authenticity of a document. The differences are due to the radically different nature of paper based documents on the one hand and digital documents on the other. In paper-based transactions a document consists of text printed as ink on a piece of paper, where the text represents the information and the paper represents the storage medium. In this way the information and the storage medium are inseparable. The

* Appears in the proceedings of AUUG2002, Melbourne, 4-6 September 2002.

† The work reported in this paper has been funded in part by the Co-operative Research Centre for Enterprise Distributed Systems Technology (DSTC) through the Australian Federal Government's CRC Programme (Department of Industry, Science & Resources)

validity of a paper-based document is authenticated by a signature written in ink on the same piece of paper. The signature serves as evidence of the signer's agreement to the text on the paper, and the only instruments between the signer and the document are optional glasses for better viewing the text and a pen for applying the signature, both of which can be considered reliable. For digital signatures all of this changes. Documents are immaterial because the information is represented by logical bits that can be stored on, and copied to, any suitable electronic medium, and they only become meaningful to humans when represented through an analogue physical medium such as a computer screen or a printout. The validity of a digital document is authenticated by verifying that an immaterial digital signature logically matches the already immaterial document. Because a digital document in its immaterial form can not be observed directly by the signer the digital signature can only serve as evidence of the signer's agreement to some analogue representation of the document although it is usually assumed that it represents the signer's agreement to the immaterial document itself. Highly complex instruments are now needed not only for viewing the document but also for producing the digital signature. Assessing the consistency and reliability of these instruments is beyond the reach of most signers.

The computer platform is the most complex element in the chain between a digital document and its analogue representation. We will not discuss platform integrity in detail, but simply mention that it is a fundamental problem, not only for correctly generating and verifying digital signatures, but for IT security in general. If the platform is not secure, then no application running on top of it can be made secure. See e.g. Loscocco *et al.* (1998) [10] for a general discussion of this topic. A more specific description of attacks on digital signatures through attacking the platform integrity can be found in Cremers *et al.* (2001) [15]. A description of some integrity vulnerabilities in the framework for digital certificates and authentication on the Web is described in Jøsang *al.* (2001) [9] and Redhead & Povey (1998) [14]. A discussion about the binding between digital documents and digital signatures can be found in McCullagh et al. (2001) [12].

This paper focuses on principles for encoding and representing digital documents, such as XML, HTML, ASN.1 and font type specifications. The flexibility that is deliberately built into these standards can make the representation and interpretation of the documents inconsistent and unreliable. When digital documents are being signed it can therefore be difficult to determine the exact content to which the digital signature applies.

2 Digital Signatures on XML Documents

2.1 False Positives and False Negatives in XML

The authors of the draft XML Signature Syntax Processing specification (Bartel *et al.* 2001) [2] are aware of potential problems when digitally signing XML formatted documents. Not only can the same XML document look different in two different applications, but different XML documents can look the same in the same application. When an XML digital signature is applied it can therefore be difficult to know exactly what is being signed. A distinction can be made between syntactic form and the semantic contents of XML documents. An important question to be answered in this context is what determines whether two different XML documents can be considered semantically equivalent, depending on their syntactic form and context.

The first case to consider is when two XML documents are semantically equivalent despite having different syntactic form. This can lead to false negatives, i.e. meaning that two documents that are semantically equivalent are not recognised as such by a human or an application. This is possible because humans are used to thinking that different form implies different meaning, and applications are designed to only make distinctions based on form. The use of canonicalisation can assist in avoiding

false negatives.

Canonicalisation (Boyer 2001) [3] is a transform whereby XML documents are transformed into a canonical syntactic form. The idea is that two XML documents having the same canonical form can be considered semantically equivalent within a given application context. However, canonicalisation does not mean that two documents are semantically equivalent *if and only if* their canonical form is identical ie. there can be cases within particular applications where two documents are semantically equivalent even though they have different canonical forms.

The second case to consider is when two XML documents are semantically different despite having the same syntactic or canonical form. This can lead to false positives, i.e. when two documents are wrongfully seen as semantically equivalent by a human or an application. This is possible because the meaning of an XML document can depend on external element type definitions (i.e. the schema, DTD, or natural language description) that are associated with the XML document, but are not part of the signature. If any of these change, the meaning of the same XML document can also change. For example, changing the DTD of an XML document does not effect its canonical form, so in the case of XML signatures, the digests will still match and the signature will verify, but the document could now have an entirely different meaning.

When an XML document is digitally signed, the signer's consent expressed by the signature applies to the semantic meaning of the document, whereas the digital signature mechanisms itself is an automatic process that ignores the meaning and only applies to the syntactic form. A distinction can thus be made between what the signer believes is being signed and what is physically being signed. Whenever false positives and negatives can occur, digital signatures are vulnerable to attack. The example below illustrates how a case of false positive can be exploited for fraudulent purposes.

2.2 Example A: Same XML but Different Meaning

This example illustrates how modifying the `ATTLIST` can change the meaning of an XML document. In the example, the author of the sample XML is a disgruntled screenwriter, and for the next episode of the http://www.etsi.org/getastandard/home.htm soap that he is working on he produces a plot outline like this:

```
<!DOCTYPE Soap [
  <!ELEMENT Character ... >
  <!ATTLIST Character
      name     ID    #REQUIRED
      nickname CDATA #IMPLIED
  >
]>

<Soap>
  <Characters>
    <Character name="alice"> ... </Character>
    <Character name="bob">   ... </Character>
    ...
    <Character name="susan" nickname="alice"> ... </Character>
    <Character name="tim"   nickname="bob"  > ... </Character>
  </Characters>
  <Actions>
    <Kill killer="#alice" victim="#bob" />
    <!-- (Uses URI fragments but could also use IDREF attributes) -->
  </Actions>
</Soap>
```

and gets sign-off from the series producer. (For concreteness, say he signs the entire document using

an enveloped signature, and puts the signature just before the closing "`</Soap>`"). Now the author modifies the `ATTLIST` declaration, changing the `ID` attribute from "`name`" to "`nickname`", and end up with

```
<!DOCTYPE Soap [
  <!ELEMENT Character ... >
  <!ATTLIST Character
      name     CDATA #REQUIRED
      nickname ID    #IMPLIED
  >
]>

<Soap>
  <Characters>
    <Character name="alice"> ... </Character>
    <Character name="bob">   ... </Character>
     ...
    <Character name="susan" nickname="alice"> ... </Character>
    <Character name="tim"   nickname="bob"  > ... </Character>
  </Characters>
  <Actions>
    <Kill killer="#alice" victim="#bob" />
    <!-- (Uses URI fragments but could also use IDREF attributes) -->
  </Actions>
  <Signature xmlns="..."> ... </Signature>
</Soap>
```

This still verifies, but now Tim dies instead of Bob, which isn't the outcome that the series producer thought he was signing.

2.3 Avoiding False Positives

Digital signatures on XML documents will be of limited value if the type of vulnerabilities illustrated in the previous example can not be avoided. In Bartel *et al.* (2001) [2] it is recommended that XML documents are canonicalised prior to digital signing, and that element type definition be signed together with the document.

Canonicalisation prior to digital signing is a transform mechanism with the purpose of avoiding false negatives. Including element type definitions in the digital signature would be a mechanism to avoid false positives, however this is not a requirement for XML signatures, therefore making them (in a minimal required state) vulnerable to the type of attack illustrated in Sec. 2.2.

These definitions could be included in the signature by either adding a reference to an external DTD or a copy of an internal DTD before signing. Defining the element types using XML schema (as opposed to DTD) is even more effective, since the ability to force a 'malicious' DTD upon a recipient is not as trivial when using schema, due to the way schemas are referenced.

Only by requiring identical element type definitions in addition to canonical form can this equivalent semantical contents be guaranteed.

3 Digital Signatures on ASN.1 Documents

3.1 False Positives and False Negatives in ASN.1

ASN.1 documents have a particular syntactic form defined by the ASN.1 standard [6]. Associated with every ASN.1 document is also a semantic content which is determined by how the document is inter-

preted by humans or applications. In this context it is important to find out what determines whether two different ASN.1 encoded documents can be considered semantically equivalent, depending on their syntactic form and context. This gives an indication of whether false positives and false negatives can occur, and thus whether digital signatures on ASN.1 documents are vulnerable to attacks.

The first case to consider is whether two ASN.1 documents can be semantically different despite having the same syntactic form. This can lead to false positives, i.e. when two documents are wrongfully seen as semantically equivalent by a human or an application. ASN.1 documents do not depend on external definitions in the same way XML documents do. For XML, external elements are explicit, whereas for ASN.1 they are implicit by assuming that everyone share a common understanding of the encoding. Under that assumption, two ASN.1 documents that are syntactically equal are thus guaranteed to be interpreted equivalently by humans or applications, thus eliminating the possibility of false positives.

The second case to consider is whether two XML documents can be semantically equivalent despite having different syntactic form. This can lead to false negatives, i.e. meaning that two documents that are semantically equivalent are not recognised as such by a human or an application. This is possible because humans are used to think that different form carries different meaning, and applications are designed to only make distinctions based on form.

There are various encoding rules for encoding the same (semantic) information, (i.e. BER, CER, DER [7] and PER [8]), opening the possibility of false negatives. The following example describes how this vulnerability can be exploited.

3.2 Example B: Failed Revocation of Digital Certificates

A particular security vulnerability described by Manger 2001 [11] exists in the 3GPP (Third Generation) Mobile Execution Environment (MExE) specification [5] relating to certificate revocation lists (CRLs). A detailed analysis of the vulnerability and a suggested solution has been sent to various parties involved with the MExE specification development (at T-Mobile, Lucent, Motorola & Vodafone).

MExE defines a certificate configuration message (CCM) that can act like a certificate revocation list (CRL) – listing certificates that must be disabled. The CCM CRL identifies a certificate by its ”fingerprint” – a hash of the complete certificate. [Note: this quantity is called a ”thumbprint” in Microsoft's certificate tools.] It is possible (and easy), however, to modify the encoding of the unsigned portions of a certificate to change the fingerprint without invalidating the signature. For instance, use indefinite-length BER for the outer wrapper, instead of DER. If the CRL issuer and relying party (MExE device) have different encoding of the same certificate the fingerprint in the CRL will not match the fingerprint calculated by the MExE device so the certificate will not be disabled when it should be.

Manger 2001 [11] describes a number of different encodings of a single certificate and each has a different fingerprint. All are (legitimately) accepted by basic ASN.1 runtime libraries. Most are (legitimately) verified as correct by Windows NT certificate validation software.

3.3 Avoiding False Negatives in ASN.1

The vulnerability in [5] exists because the hash of a received certificate can be taken of **any** particular encoding such as DER or BER, rather than of a **canonical** encoding such as CER and to a certain extent DER. When the specification allows hashing of non-canonical encodings, it is evident that hash values may not match. One possible solution is to only hash what is being signed. Another solution is to always compare hash values of ASN.1 documents based on canonical encoding such as DER and CER. This will effectively eliminate the type of false negatives describe in the above example.

4 Manipulation of Font Types

4.1 The Role of Font Types for Correctly Viewing Digital Documents

The specification of how the information in a document, encoded as immaterial bits, is displayed as an analogue image on a computer screen or a printout, is crucial for correctly bringing the meaning of a document to the attention of a human observer, for example prior to applying a digital signature. Central to this discussion lies the distinction between characters and glyphs. Characters are codes assigned by the Unicode standard [16] which represent the smallest semantic units of language, such as letters. Glyphs are the specific forms that those characters can take, as defined by the font type. One character may correspond to several glyphs providing alternate forms of the same letter within the same font type. This can be achieved by additional layout features of a given typesetting standard such as e.g. OpenType [1]. One glyph can also represent multiple characters, as in the case of the ”ffi” ligature, which corresponds to a sequence of three characters: f, f and i.

Although a character is usually associated with some graphical form, it is not up to the Unicode standard to dictate that form in every detail, the standard only indicates one typical form for each character. Theoretically it would be possible to define characters as sounds instead of graphical forms, in the same way as it is possible to play sequences of characters as audible words instead of displaying them as written words. The actual shape of a character in a particular application is defined by the specification of the font type used by the application and the font interpreter for the particular analogue device, such as a computer screen or a printer. Ultimately, the interpreter generates a pixel matrix for each character as a function of the shape and size of the glyph and the resolution characteristics of the analogue output device. The correspondence between the bits that constitute a character code and the analogue display of a glyph is conceptually illustrated on Figure 1 below.

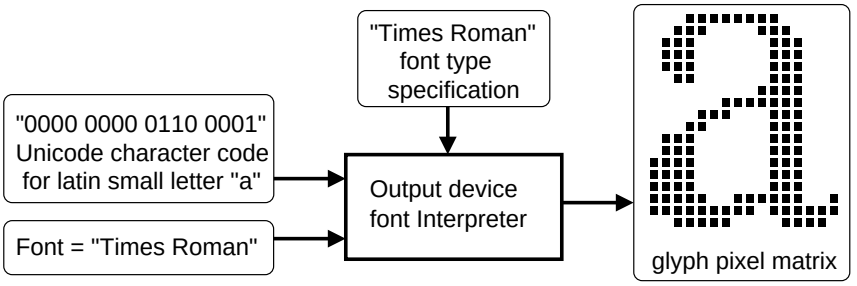

Figure 1: Correspondence between character code and glyph

Depending on the document type and application, digital documents can be defined with or without specifying the font. In case the font is not specified, the application will use some default system font when displaying or printing the document. In case the font is specified, the application will try to use that font when displaying or printing, but if it is unavailable on the system, will substitute the font with a default or a similar font. When substitution takes place, it can happen tacitly or with a warning to the user, depending on the application and configuration. In the standard configuration of Microsoft Word for example, no warning is given. By looking at the formatting toolbar however, it is normally possible to find out what font type was originally applied to a character or a block of text. But the formatting toolbar does not necessarily give reliable evidence because utilities exist for MSWindows (e.g. *TrueType Font Namer* [13]) to change font type names, so that in practice any font type can have any name. Although there are a limited number of commonly used font types there can in theory be an arbitrary number of different font types. Within the same MSWord document for example, each character can have a different

font type.

The above discussion indicates that the font name and the font type specification are variables that can change the optical appearance of digital documents. This flexibility can of course also be misused with malicious intent, in particular in case of digital signatures. We will describe two possible scenarios for how this can happen.

4.2 Example C: Substituted Fonts

In this scenario, Clark prepares a digital document with the following contents displayed in Helvetica font:

On 24 October 2001, Alice borrowed from Clark the sum of $1000.

However, Clark defines a new font called "Helvetika" which is similar to Helvetica in style, but for which the glyphs for "A", "i", "c" and "e" have been interchanged with the glyphs for "C", "a", "r" and "k", and installs this font on his computer. Clark then applies this new font to the words "Alice" and "Clark" with the result that when viewed on his computer the document looks like:

On 24 October 2001, Clark borrowed from Alice the sum of $1000.

Clark then borrows $1000 from Alice and digitally signs the document. Alice is satisfied by visual inspection of the document and verification of the digital signature on Clark's computer. Alice copies the digitally signed document to her floppy disk as evidence for Clark's debt to her. When Alice tries to prove her case and displays the digitally signed document on a court room computer the font Helvetika is replaced with Helvetica or some other default font, with the result that the evidence becomes valueless.

In this scenario, it would have been possible for Alice to detect that there was something wrong had she checked the fonts used in the document before signing. For a small document like in this example that might be feasible, but it soon becomes impractical when the size increases to more than a few lines. The court can also interpret the unknown font type Helvetika as evidence that the document could have been manipulated with malicious intent.

4.3 Example D: Changed Font Names

In the previous example it was theoretically possible to detect the fraud. This example shows how it is possible to avoid detection by hiding the fact that font substitution takes place.

In this scenario, Clark prepares a digital document with the following contents when displayed in Helvetica font:

On 24 October 2001, Clark borrowed from Alice the sum of ¥1000.

Clark then creates a font type which is similar to Helvetica in style, but for which the glyphs for "¥" and "$" are interchanged. Clark calls this new font type "Helvetica", but we will call it "Helvetica′" in order to distinguish it from the original one. Clark substitutes the original Helvetica font type with Helvetica′ on his computer, with the result that the document looks like:

On 24 October 2001, Clark borrowed from Alice the sum of $1000.

Clark then borrows $1000 from Alice and digitally signs the document. Alice is satisfied by visual inspection of the document and verification of the digital signature on Clark's computer. Alice copies the digitally signed document to her floppy disk as evidence for Clark's debt to her. When Alice tries to prove her case and displays the digitally signed document in a court room the font Helvetica′ is replaced with Helvetica, with the result that the evidence indicates a debt of ¥1000 instead of $1000.

In this scenario Alice has no way of verifying that the font type has been manipulated before signing, because Clark's new font carries the name she expects to see. The court is equally unable to find indications of malicious manipulation of the document.

4.4 Controlling the Font Type

A simple countermeasure against substituted font types is to hash the font type specification of all font types used in the document to be signed, and to include the hash in the digital signature, and to disallow substitution of font types. By caching the hash value of each font type specification performance deterioration will be minimal. Disallowing font type specification will only cause a minimal reduction in the flexibility of document exchange. In practice, it will mean that exotic and customised font types in digitally signed documents should be avoided because they are likely to make signature verification impossible.

5 Digital Signatures on Graphical Features

Digital documents can contain more than just text. Common examples of additional graphical features are pictures, drawings, table formatting and background colour. The correct and consistent rendering of such features is crucial for the meaning of digital documents. If graphical features are displayed differently in different applications, then the meaning is likely to change, making it meaningless to digitally sign such documents. The example below shows how inconsistent handling of table tags in HTML can make the same table look completely different.

5.1 Example E: Inconsistent Handling of HTML Table Tags

In this scenario it is assumed that three building contractors have submitted a quote for building an office building for a company. The quotes are: "Alice Architects and Builders": $800,000, "Bob Building Contractors":$900,000, and Clark Constructions": $1,000,000. The company managers who evaluates the quotes is satisfied with the qualifications of all three contractors, and decides to list his preference as a function of price. The manager asks the company's web editor to create a table with the building contractors listed according to price. The manager digitally signs the html page seen in Fig.2 and submits it to the company's procurement department for further processing.

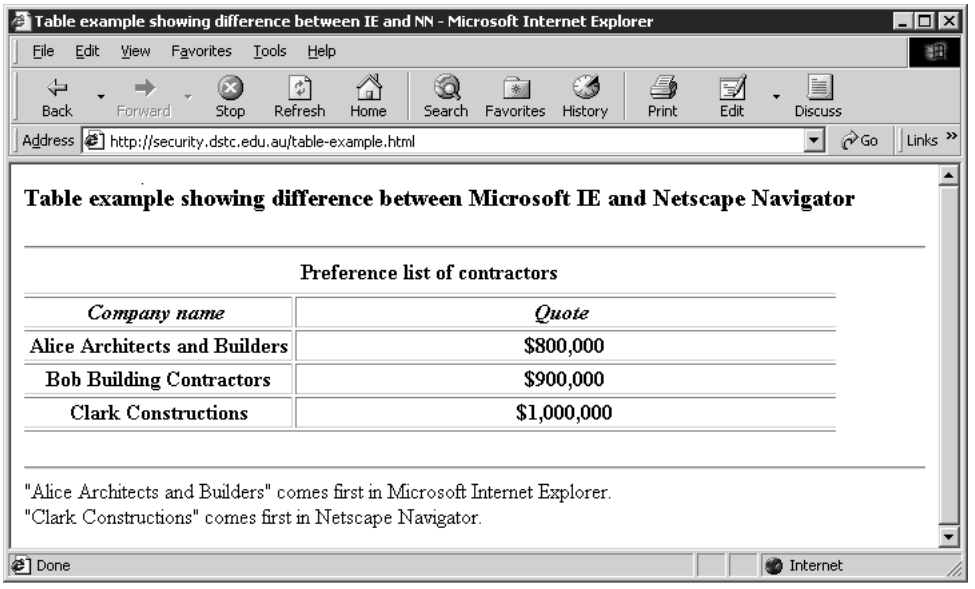

Figure 2: Viewing the table in Microsoft Internet Explorer

What the company manager does not know is that the web editor is a close friend of Clark, and therefore will try to make “Clark Constructions” win the contract. The web editor knows that the manager uses Microsoft Internet Explorer, whereas the procurement department uses Netscape Navigator. The Web editor encodes the HTML table so that “Alice Architects and Builders” gets highest preference when viewed with Microsoft Internet Explorer, and “Clark Constructions” gets highest preference when viewed with Netscape Navigator. The figures below show what the tables look like when viewed in Netscape Navigator.

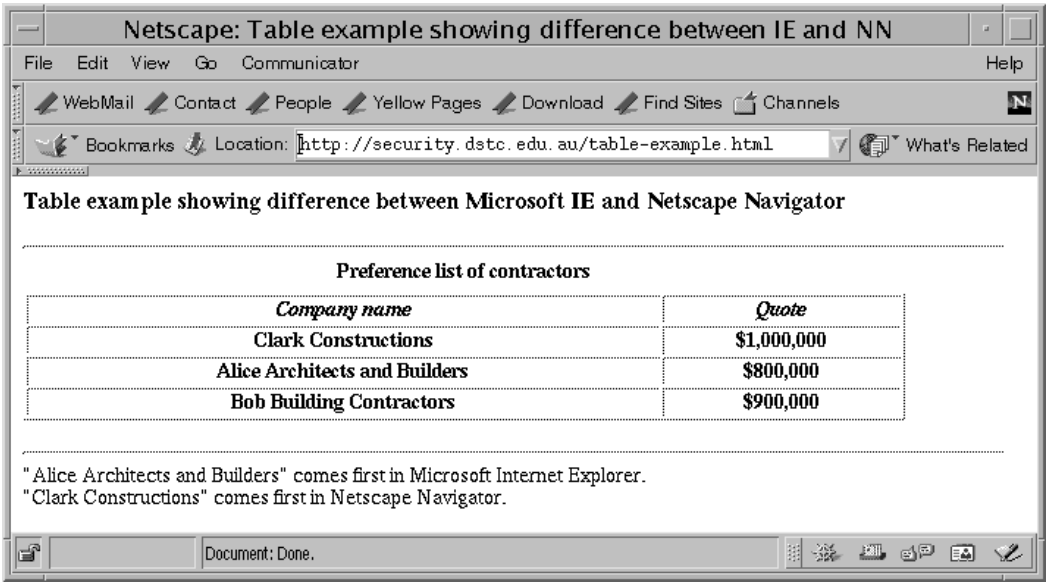

Figure 3: Viewing the table in Netscape Navigator

Figure 2 shows what the managers sees and signs. Figure 3 shows what the procurement department sees. Note that that the HTML code is identical in both cases.

This is possible because the HTML tag `tfoot` is handled inconsistently. Microsoft Internet Explorer always creates a row at the end of the table whereas Netscape Navigator creates a row at the point where the tag appears in the HTML code. Thus by encoding “Clark Constructions” with the tag `tfoot` just after the table head, that row will appear to be the last table entry in Microsoft Internet Explorer, and the first table entry in Netscape Navigator. The complete HTML code for this example is shown below.

```
<!DOCTYPE html>
<html>
<head>
<title>Table example showing difference between
       IE and NN</title>
</head>

<body bgcolor="#ffffff">
<h3>Table example showing difference between
Microsoft IE and Netscape Navigator</h3>
<hr />
<table border="1" width="90%" align="centre"
```

```
       frame="hsides" rules="groups">
  <caption><b>Preference list of contractors</b></caption>
<colgroup width="20%" align="center" valign="top">
<colgroup span="2" width="40%" valign="top">
<thead>
  <tr>
    <th><i>Company name</i></th>
    <th><i>Quote</i></th>
  </tr>
</thead>
<tfoot>
  <tr>
    <th>Clark Constructions</th>
    <th>$1,000,000</th>
  </tr>
</tfoot>
<tbody>
  <tr>
    <td align="center"><b>Alice Architects and
                          Builders</b></td>
    <td align="center"><b>$800,000</b></td>
  </tr>
</tbody>
<tbody>
  <tr>
    <td align="center"><b>Bob Building
                          Contractors</b></td>
    <td align="center"><b>$900,000</b></td>
  </tr>
</tbody>
</table>
<br>
<hr />
"Alice Architects and Builders" comes first in
Microsoft Internet Explorer.<br>
"Clark Constructions" comes first in Netscape Navigator.
</body>
</html>
```

5.2 Avoiding Inconsistency when Rendering Graphical Features

The way graphical features are handled can be extremely complex depending on context and software implementation, and there does not seem to be a general solution for guaranteeing consistency when digitally signing documents with graphical elements.

6 Discussion

Inconsistencies regarding the representation of digital documents normally pose few problems because authors and readers usually have a positive attitude and a goodwill to understand each other. In that sense it can be argued that most digital documents can be signed with little risk of misunderstanding. Unfortunately this way of reasoning is not viable for security. Designing secure solutions requires the opposite way of thinking, and the question to be asked is: *What if somebody deliberately tries to cause misunderstanding, would he or she succeed?* The examples described in this paper shows that it is relatively easy to create confusion regarding the semantic content of digital documents and thus that it can be difficult to know what a digital signature applies to.

Section 2 and Section 3 mention the use of canonical form for uniquely representing digital documents in XML and ASN.1. As mentioned in Section 2 however, canonicalisation does not guarantee that semantically equivalent documents necessarily must have identical canonical form. It is extremely difficult to completely avoid ambiguity in the specifications of canonical forms, making it almost impossible for software vendors to write implementations that produce exactly the same results. Canonical form can therefore only give an indication, and not a guarantee, that two documents are semantically equivalent.

Some of the examples described above had the purpose of creating specific alternative meanings to digital documents. Another form of attack could be to simply create ambiguity about a document's meaning, and thereby make the document unacceptable or inadmissible as evidence. An attacker could for example hide ambiguity in a digital contract in order to have the freedom at a later stage to claim that the contract is invalid on that basis and thereby be liberated from the contractual obligations.

A simple conclusion to be drawn from our analysis is that the more flexible and open the framework for handling digital documents is, the more complex the correspondence between the digital form on the one hand and it's interpretation or analogue representation on the other becomes. It is in this space that the vulnerabilities we have described are found and that attacks can be mounted.

A distinction can be made between machine interpretation and human interpretation of documents. The first category applies to XML and ASN.1 documents that can be interpreted, digitally signed and verifued by software and hardware without direct human intervention. In this case it should in principle be possible to completely avoid false positives and false negatives. The failure of XML Digital Signature and the MExE specifications to achieve this can only be attributed to specification flaws that can and should be rectified.

In the second category where human interpretation of digital documents depends on visual inspection, ambiguity can be avoided by signing the bitmap that constitutes the analogue graphical representation of a digital document. This applies to all examples mentioned in the previous sections, including XML and ASN.1 documents. As mentioned in Bartel *et.al* (2001) [2] this would result in data that are difficult for software applications to subsequently manipulate. A possible solution is for the signer to always archive an analogue image of all documents he or she signs, and to include the hash of that image in the digital signature. In that way it is possible to refer to what the signer really saw at the moment the signature was applied, should it be required at a later stage.

7 Conclusion

The term "digital signature" is a metaphor that can make people falsely believe that it is equivalent to handwritten signatures. However, it should be seen as a new paradigm proper to computer systems rather than treating it equivalent to handwritten signatures. The original form and analogue representation are completely separate phenomena for digital documents, whereas paper based documents have both combined into one. Whenever the meaning of a digital document can only be accessed by viewing an

analogue representation of it, the consent expressed by the digital signature should only apply to what the signer sees, i.e. to its analogue representation, and not to the immaterial bits that constitute the digital document in its original form.

Acknowledgements

Thanks to Thomas Maslen from Wedgetail Communications for providing the XML code for the soap plot in Section 2.2. Thanks to Andrew Mich from Telstra Research Laboratories for providing the example about ASN.1 in Section 3.2.

References

[1] Adobe. *OpenType User Guide*. Adobe, 2000. URL: http://www.adobe.com/type/browser/pdfs/OTGuide.pdf (visited 18 September 2001).

[2] Mark Bartel et al. *XML-Signature Syntax and Processing - W3C Proposed Recommendation 20 August 2001*. W3C (World Wide Web Consortium), 2001. URL: http://www.w3.org/TR/2001/PR-xmldsig-core-20010820/.

[3] John Boyer. *Canonical XML, Version 1.0 - W3C Recommendation 15-March-2001*. W3C (World Wide Web Consortium), 2001. URL: http://www.w3.org/TR/2001/REC-xml-c14n-20010315.

[4] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[5] ETSI. *ETSI TS 123 057 V4.2.0 (3GPP TS 23.057 V4.2.0), UMTS Mobile Execution Environment (MExE) Functional Description, Stage 2*. European Telecommunications Standards Institute, June 2001. URL: http://www.etsi.org/getastandard/home.htm (visited 01.10.2001).

[6] ISO. *IS 8824-1,2,3,4. Abstract Syntax Notation One (ASN.1). Part 1: Semantic model, Part 2: Information object specification, Part 3: Constraint specification , Part 4: Parameterization of ASN.1 specifications*. International Organisation for Standardization, 1998.

[7] ISO. *IS 8825-1. ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*. International Organisation for Standardization, 1998.

[8] ISO. *IS 8825-2. ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)*. International Organisation for Standardization, 1998.

[9] A. Jøsang, P.M. Møllerud, and E. Cheung. Web Security: The Emperors New Armour. In *Proceedings of the European Conference on Information Systems (ECIS2001)*, Bled, Slovenia, June 2001.

[10] Peter A. Loscocco et al. The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments. In *Proceedings of the 21st National Information System Security Conference*, pages 303–314. NSA, October 1998.

[11] James Manger. Vulnerability in 3GPP (Third Generation) Mobile Execution Environment (MExE) specification relating to certificate revocation lists (CRLs). Defect report to 3GPP standartisation working group, 2001.

[12] Adrian McCullagh, William Caelli, and Peter Little. Signature Stripping, A Digital Dilemma. *Journal of Information, Law and Technology*, 6(1), 2001.

[13] UniTech / MyTools. TrueType Font Namer. URL: http://www.mytools.com/fontnamer.html (visited 20 September 2001.

[14] Tim Redhead and Dean Povey. The Problems With Secure On-line Banking. In *Proceedings of the XVIIth annual South East Asia Regional Confereance (SEARCC'98)*, July 1998.

[15] Adrian Spalka, Armin B. Cremers, and Hanno Langweg. The fairy tale of 'What You See Is What You Sign - Trojan Horse Attacks on Software for Digital Signatures. In *IFIP Working Conference on Security and Control of IT in Society-II (SCITS-II)*, Bratislava, Slovakia, June 2001.

[16] UNICODE. *The Unicode Standard, Version 3.0*. Addison Wesley Longman Publisher, isbn 0-201-61633-5 edition, 2000. URL: http://www.unicode.org/unicode/standard/standard.html (visited 20 September 2001).