

Improper Utilization of Digital Signature Technology

John Boyer

UWI.Com

Introduction

Basic digital signatures with public key cryptosystems [CLR]

Signature = Encrypt(hash(M), Signer.PrivateKey)

Verified = hash(M) == Decrypt(Signature, Signer.PublicKey)

Problem

Depends on security of public key delivery to verification

Solution

PKI and CAs

Another Problem...

What are some of the things that a signature does? [ABA]

- **Message authentication: offered by hash, protected by encryption**
- **Signer authentication: offered by key pair, protected by PKI**

Why is Signer S signing M?

- **To give M to another party, the verifier V.**
- **M transfers information from S to V**
- **M can represent a transaction between S and V**
- **Once V authenticates S and M, V will perform the transaction represented by M.**

Problem: What if M is not a good representative of the transaction? [BB]

Example: Signing an HTML submission

Field1=500&Field2=50000

I agree to purchase 500 steel beams for 50000 US dollars.

I agree to pay 500 US dollars for 50000 US dollars of life insurance coverage.

I agree to a Cdn \$500 deductible and a Cdn \$50000 premium on this building insurance.

I agree to donate 500 computers and 50000 floppy disks to this charity.

What's going on?

Same HTML submission so same signature validates all of these statements

- **Signer obligated to pay different amounts in different currencies**
- **Signer receives different goods**
- **The last does not even involve a monetary payment, and no goods are received by signer.**

Commonalities: 500, 50000, and the order in which they appear.

These examples are oversimplified, but show that something has gone badly wrong in this usage of digital signature technology.

Whom does the technology disservice?

- **Technology disserves people when it applies digital signatures to messages that are poor representatives of the transaction are a disservice.**
- **We often think of the signer being disserved since the recipient can exploit a security hole to change a transaction.**
- **However, even the recipient is disserved since the recipient relies on the technology to prove the signer's authorization of a specific transaction before shipping goods with an invoice attached.**

Why did things go so badly?

Context has been lost. We have answers but no proof of questions.

So, what if the questions were added to the text that got signed?

No proof that the questions were in a particular order on the page, so any wording ambiguity can be exploited. So, we also need the positions of the questions and answers.

What about adding the text of the 'fineprint'?

No proof that the fine print wasn't the same color as the background or that the font was large enough.

In general, we need to capture the whole context including the questions, answers, as well as positions, colors, fonts and so on. "What you see is what you sign."

AIIM Guidelines

The Association for Information and Image Management's (AIIM) Performance Guideline for the Legal Acceptance of Records Produced by Information Technology Systems provides the following guidelines:

The following shall appear with sufficient clarity so that each can be recognized:

- **Individual letters, numbers and symbols;**
- **Combinations of letters, number, and symbols forming words or sentences;**
- **Graphics, such as signatures, logos, pictures, etc.;**
- **Sounds;**
- **Other features of records such as color, shape, texture, etc. that relate to the content of the information.**

These guidelines illustrate the need for transaction records to contain not only the content of an agreement, but the context and structure as well. [AIIM]

Signing HTML (and XFDL)

The result so far ...

From browser get access to HTML parse tree

- M=Form markup with user data inserted
- Digest M, Base 64 encode, and store result
- On server side, reconstruct full HTML form from HTML submission
- Use Digest to verify

This is what [XFDL] does except

- Form kept as one unit at all times
- Intended for complex business/government forms

Conclusion: Signing full markup = better message M

Signing XML

XFDL and HTML are specific vocabularies for XML

The new web standard for representing information is XML.

We should be signing XML markup. [DSIG]

XML is a grammar; it has virtually no lexicon (and hence no semantics).

It is impossible to assess whether a message M is a 'good representative' without knowing the semantics (and hence the lexicon) of the XML being signed.

We will examine hypothetical scenarios to see what can go wrong when signing XML.

Intro to XML

- XML wraps start and end tags around data, marking it up
- The tags give meaning to the data
- The start tags can have attributes that add further meaning

```
<SpreadSheet>
<A>
<c1 format="string">Total Sales:</c1>
<c2 format="dollar">$4062.44</c2>
</A>

... <!-- More -->

</SpreadSheet>
```


<c1 format="string">Sales Tax Collected:</c1>

<c2 format="dollar" compute="0.07*A2">\$279.21</c2>

<C>

<c1 format="string">Vendor Deduction:</c1>

<c2 format="dollar" compute="0.033*B2">\$9.21</c2>

</C>

<D>

<c1 format="string">Net Tax Due on Sales:</c1>

<c2 format="dollar" compute="B2 - C2">\$270.00</c2>

</D>

<E>

<c1 format="string">Tax Paid on Goods:</c1>

<c2 format="dollar">\$57.00</c2>

</E>

<F>

<c1 format="string">Total Amount Due:</c1>

<c2 format="dollar" compute="D2 - E2">\$213.00</c2>

</F>

Inclusion List Method and Code Signing

Inclusion List: each list entry identifies single element or attribute

Code Signing (why): recipient wants “government approved” spreadsheet

Code Signing (how):

- Want the strings, the formats and the computes
- Don't want the values since user can change those

```
<Signature>
<Manifest>
  <Object>A/c1</Object>
  <Object>A/c2/@format</Object>
  <Object>B/c1</Object>
  <Object>B/c2/@format</Object>
  <Object>B/c2/value/@compute</Object>
  ... <!-- C, D, F like B and E like A -->
</Manifest>
<Certificate>...<Certificate>
<Value> Iamabase64encodedshaldigest=</Value>
</Signature>
```

What's Wrong With This Message?

```
<c1 format="string">Total Sales:</c1>
format="dollar"
<c1 format="string">Sales Tax Collected:</c1>
format="dollar"
compute="0.07*A2"
<c1 format="string">Vendor Deduction:</c1>
format="dollar"
compute="0.033*B2"
<c1 format="string">Net Tax Due on Sales:</c1>
format="dollar"
compute="B2 - C2"
<c1 format="string">Tax Paid on Goods:< /c1>
format="dollar"
<c1 format="string">Total Amount Due:</c1>
format="dollar"
compute="D2 - E2"
```

Problems Depend on Processing Rules

Original idea was to put digest of each Object into Object then have signature over Manifest

- **Element order in document can change without breaking signature.**
- **Digesting messages that are too small.**

With message in document order, still lost ancestor element information.

- **What does the format attribute format?**
- **What does the compute attribute compute?**
- **In which row is c1?**

This example begins to work when the manifest is added to the signed message, but only because the document processing rules are so simple.

Changes to a document can easily cause the verified signature to have little or nothing to do with what the application shows the user.

Other Omission Scenarios

- **Multiple Signatures**
- **Office use only section of a form**
- **Omit parts not related to application (like data views for backend programs)**

Why are these failures happening?

Because restriction to inclusive logic typically results in poor representative messages.

Why?

- Markup is meant to put tags around information to give more meaning to the information. The tags around information are the 'ancestor' elements.
- Elements can derive information from ancestor tags and attributes.
- The Inclusion method forces one to omit ancestor tags and attributes when one really wants to omit certain descendant nodes in the parse tree, like the content representing the data value.
- For designing an XML signing method, it does not matter if the XML to be signed was not designed the way we prefer; the only thing that matters is whether the XML is well-formed. If it is, then the XML signing method must provide a way to secure it.

What does it take to construct a good representative?

Consider document D as a 'universal set'.

Suppose we want to sign a subset S of elements from D .

Suppose that elements in S have semantic dependencies on elements in another set S' .

The union of S and S' is a better representative.

But the elements in S' may have further dependencies on another set S'' of elements in the document.

The union of S , S' and S'' is an even better representative message.

In general, we are talking about an operation that graph theorists call transitive closure of S .

What does it take to compute transitive closure for XML?

XML has (virtually) no vocabulary and semantics of its own, so it could be the case that the semantic dependencies span every edge in a complete graph of the XML elements.

So, we must traverse the entire document to decide which elements are in and which are not in the transitive closure of the set S of elements we want to sign.

But whose rules will make the decision for each element?

Only the person writing the document knows the XML vocabulary being used, so only the document author can describe the element test

Marker Method

Put special markers around regions that should be signed

```
<?begin S?>
<SpreadSheet>
<A>
<c1 format="string">Total Sales:</c1>
<c2 format="dollar"><?end S?>$4062.44<?begin S?></c2>
</A>
...
</SpreadSheet>
<?end S?>

<Signature>
<Object Region="S"/> <!-- empty means this doc -->
<Certificate>...</Certificate>
<Value>Iamabase64encodedshaldigest=</Value>
</Signature>
```

Advantages:

- Can retain ancestor info and document element order

Disadvantages:

- May not have write access to indicated Object
- Object may already have S markers in it
- Can't put markers around subset of attributes
- Arbitrary material can be added between 'end' and the next 'begin'

Element Test Method

Write a Boolean test expression that will be applied to each node of the parse tree in document order. If the expression returns true, then the element is kept, and if it returns false, the element is omitted.

Example:

```
<Signature>
...
<Transform>
/descendant-or-self::node()[
    not(self::text() and parent::c2 and ancestor::A[2])
and   not(self::text() and parent::c2 and ancestor::B[2])
and   not(self::text() and parent::c2 and ancestor::C[2])
and   not(self::text() and parent::c2 and ancestor::D[2])
and   not(self::text() and parent::c2 and ancestor::E[2])
and   not(self::text() and parent::c2 and ancestor::F[2])
]
</Transform>
...
</Signature>
```

Element Test Method

Notes

- The test is applied to an element's descendants even if the element itself is omitted.
- It is a node test not a subtree test.
- Use ancestor-or-self on a node test to eliminate a node's descendants

Advantages:

- Can retain ancestor info and document element order
- Works without modifying target document
- Precisely expresses what can change without breaking signature

Disadvantages:

- Test expression parser is harder to implement

Document Closure

Clearly, the Marker and Element Test methods are superior since they allow more precise indication of regions to omit from the digested part of the document

Define T to be the transitive closure of S , the set of elements we want to sign.

Define $\sim T$ to be the set of elements in the document D that are not in T (i.e. $D - T$).

Document closure – an expression indicating elements that must be changed in order to finish the document.

If a digital signature covers a certain set T of elements in the document D , then $\sim T$ is precisely the set that is allowed to change.

A metric for determining the best method of transitive closure is the precision with which the method can both capture T but also restrict $\sim T$ to precisely those elements necessary to achieve document closure.

The marker method is less sensitive to changes in or additions to the document that are outside of the regions covered by the signature.

The element test method applies the node test, even to elements in regions that were omitted when the signature was created.

REFERENCES

[ABA] M. Baum & R. Schwartz. (Eds.) Digital Signature Guidelines: Legal Infrastructure for Certification Authorities and Secure Electronic Commerce. American Bar Association, Section of Science and Technology, 1996. Available at:

<http://www.abanet.org/scitech/ec/isc/dsgfree.html>

[AIIM] The Association for Information and Image Management (AIIM). Performance Guideline for the Legal Acceptance of Records Produced by Information Technology Systems, Part IV: Model Act and Rule, Part 3. Also, Proposed Model Rule for Acceptance of Records, §6. Criteria for determining acceptance of records.

[BB] B. Blair & J. Boyer. XFDL: Creating Electronic Commerce Transaction Records Using XML. Proceedings of The Eighth International World Wide Web Conference, Toronto, May 11-14, 1999. Available at:

<http://www.w8.org/w8-papers/4d-electronic/xfdl/xfdl.html>

[CLR] T. Cormen, C. Leiserson, & R. Rivest. Introduction to Algorithms. MIT Press, 1990.

[DSIG] D. Eastlake, J. Reagle, D. Solo, M. Bartel, J. Boyer, and B. Fox. XML-Signature Core Syntax and Processing. W3C Working Draft, 22 October, 1999.

<http://www.w3.org/TR/1999/WD-xmlsig-core-19991022.html>

[XFDL] J. Boyer, T. Bray, and M. Gordon (Eds.). Extensible Forms Description Language (XFDL) 4.0. W3C Note, Spetember 2, 19998.

<http://www.w3.org/TR/NOTE-XFDL>