
ORACLE®

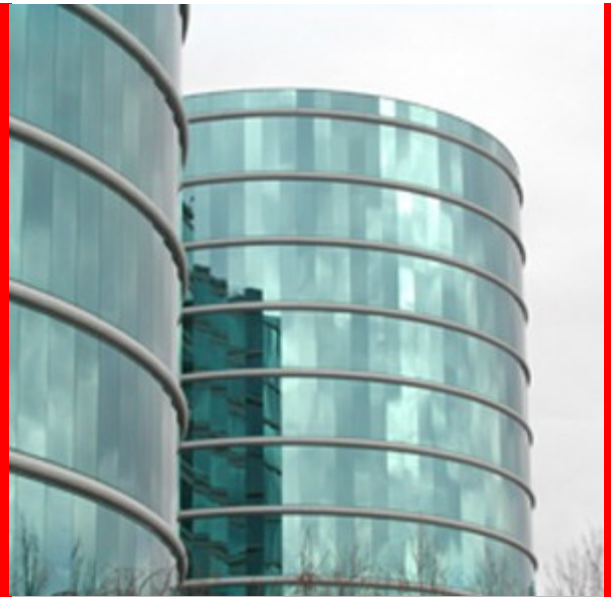
ORACLE®

Experience with XML Signature and Recommendations for future Development

Prateek Mishra, Sep, 2007

Overview

- Two main recommendations:
  - Working group should consider a “simple” profile of DSIG
    - Allow developer who isnt an expert in XML or security to use DSIG
    - No schema information required for document being signed or for any utility classes (wsu:id)
    - Restrict use of XPATH to the maximum extent
    - One-pass extraction of keys and signatures
  - Development of a streaming profile of DSIG
    - Supports a scalable implementation of DSIG
    - Pratik Datta will describe his proposal



ORACLE

Streaming Implementation of XML Signatures

Pratik Datta (pratik.datta@oracle.com)

Problem definition

- XML Signature implementations are expensive
  - NodeSets require DOM.
    - DOM expands a document 20x times
  - Transforms require a lot of temporary memory
    - Each transform's entire input and output needs to be in memory
    - Deallocation/Garbage collection is expensive

Solution - streaming

- No time benefit – only a space benefit.
  - If you had enough memory, a stream based impl would take a similar amount of time as a DOM based impl.
  - With stream – memory usage is not proportional to document size but constant (1 block size) – predictable scalability
- Space benefit implies – no disk thrashing, no excessive garbage collection, far lesser memory allocation calls
  - Indirectly giving a time benefit
- Not very useful if the document is being loaded up in a DOM anyway
  - If security logic and application logic are using the same DOM, no benefit is streaming.
  - Most beneficial in “gateway” kind of products

Streaming <> “One Pass”

- One pass is more restrictive form of streaming
  - One pass means strictly no backward references which may violate some higher level protocols – (In WSSecurity it is possible to sign objects before the signature. e.g. a secure timestamp)
- Two pass is also streaming
  - In first pass collect all signatures, tokens, keys etc. In second pass verify signatures
  - Two pass streaming does not take away any of the space benefits
  - Although the entire document needs to be present in disk/memory, It never needs to be loaded up into a DOM. Still using a constant amount of memory.
  - Very amenable for Hardware Acceleration cards which have limited amounts of onboard memory, but can read main memory one block at a time.

Streaming Requirements

- Streaming solutions have been explored before
  W. Lu, K. Chiu, A. Slominski, D. Gannon
  , “A Streaming Validation Model for SOAP Digital Signature”
- But this solution does not allow transformations. Our primary use case is WS Security, which needs to support the following transforms
  - Enveloped Sig
  - XPath Filter (limited)
  - XPath Filter 2.0 (limited)
  - STR Transform
  - Decrypt Transform (needs streaming xml encryption impl)

Proposal – xml event stream based model

- We propose a new “xml event” stream based model with a “pipeline” based transform procssing

Alternative to Nodeset
A SAX/StAX/XMLReader stream

- Fundamentally we need to change a nodeset to an “XML event stream”.
- E.g. for this document

```
<A>hello
  <B c="1"/>
</A>
```

  - When represented as a nodeset

    `A, hello, B, c`

    - A nodeset is not complete information by itself, It needs a backing DOM tree
    - Nodeset is not ordered
  - When represented as a xml event stream

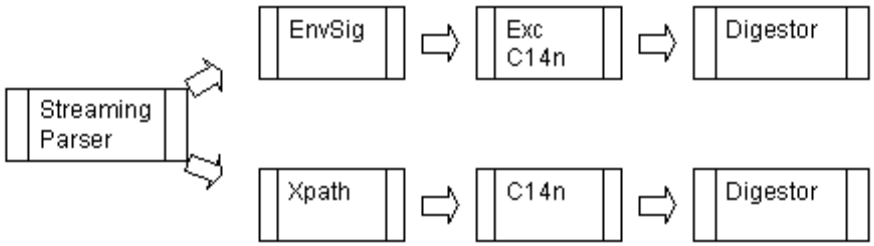
    `beginElement(A), text(hello), beginElement(B), attribute(c), endElement(B), endElement(A)`

    - XML event stream is a complete information.
    - Alternatively Attributes can come along with the beginElement event

“Pipeline” processing model with xml event stream

- Signature generation/verification will use a “pipeline”
  - One “pipeline” for each reference
  - Pipeline consists of “processing nodes” , one for each transform
  - End of the pipeline is a digestor node
  - Selection by ID or xptr is also treated a processing node

- E.g. Signature with two references
  - 1st reference has EnvSig, ExcC14n
  - 2nd reference has Xpath, C14n



Pipeline model (Contd)

- ID selection, Xpath filters and EnvSig transform nodes
  - Input xml events, and output xml events.
- C14N and STRTransform nodes
  - Input xml events and output byte buffers
- Digestor node
  - Input byte buffers and computes a digest

- Note: Each processing node DOES NOT have access to the whole document. It only knows the current node, and all ancestor nodes

Problems with this processing model

- These problems can be solved if we limit to a subset of the xml sig spec

Problem 1. Some nodesets can't be represented by xml event streams

- In general a nodeset that is subset of the document can be represented as an event stream. E.g. for this document

```
<A>foo
  <B c="1">
    <D>hello</D>
    <E>world</E>
    <F>
      <G h="2"/>
    </F>
  </B>
</A>
```

  The nodeset

```
foo, D, hello, G, h
```

  Can be represented as

```
text(foo), beginElement(D), text(hello), endElement(D), beginElement(G), attribute(h),
   endElement(G)
```

  Note: this is not valid XML because it has multiple root elements, and text outside a root element, but it is still streamable

Problem 1. Contd

- However for the same document

```
<A>foo
  <B c="1">
    <D>hello</D>
    <E>world</E>
    <F>
      <G h="2"/>
    </F>
  </B>
</A>
```

This nodeset can't be represented.

```
A, foo, c
```

That's because the attribute c is in the nodeset, but its owner element B is not.

Constraint: To solve this, we propose to profile XML Signature spec to constrain nodesets to disallow attribute nodes without corresponding parents

Problem 1. Contd

- Another kind of nodeset that can't be represented as an XML event stream involves namespace filtering.

Suppose the XML document is:

```
<n:A xmlns:n="http://foo" xmlns:n2="http://bar" >
<n:B>
  <n:C/>
</n:B>
</n:A>
```

Even though the namespaces n and n2 are defined only once, in a nodeset they are present for every descendant. i.e. the nodeset is actually

```
A, n, n2, B, n, n2, C, n, n2
```

Now a subset of this nodeset could have some namepace nodes missing e.g.

```
A, n, n2, B, n2, C, n, n2
```

In the above nodeset, B's n is missing. This is not valid XML, because B uses n. So it cannot be represented in an xml event stream. This motivates the second constraint.

Constraint: To solve this, we propose to profile XML Signature spec to constrain nodesets to disallow removal of namespace nodes that are used

Problem 2. Xpath Filtering

- As stated before a “processing node” only has the context of the current node and all ancestor nodes, but not the entire document.
  - This means that an XPath filter processing node has to operate on a very limited set of information, **We impose the following constraints in the profile**
    1. Can only use the “ancestor”, “parent”, “self”, “attribute” and “namespace” axes. (cannot use“child”, “descendant”, “following”, “preceding”, “following-sibling”, “preceding-sibling” axes). Also attributes and namespaces can only be off the self axis, not parent or ancestor.
    2. Cannot use “string-value” of element nodes or root nodes as “string-value” involves looking at all the descendant nodes and concatenating their text values. If an element has only one text node child, a streaming parser might return the the text as separate chunks, so it makes it very difficult for implementations to compute this.

Problem 2. Contd

- These constraints on XPath expressions are not too limiting. It is still possible to
  - choose a SOAP:Body element
  - choose a SOAP:Header element that has a particular actor
  - choose all elements with a particular name
  - choose an element with a particular attribute value
  - exclude all elements with a particular name or attribute value
  - do EnvSig type of xpath

- But it is not possible to do something like this
  - choose the 3rd line item. (This requires siblings context)

Problem 2. Contd

XPath Filter 2.0 is easy in DOM, because the Xpath expressions need to be evaluated only once, whereas in XPath Filter 1.0, they need to be executed for every node in the nodeset.

- But they are more difficult in streaming because the entire document is never available at once, Instead the implementation would need to convert the XPath Filter 2.0 to an XPath Filter 1.0, and evaluate that.
- Conversion from XPath Filter 2.0 to XPath Filter 1.0 is extremely difficult, so we need a very **constrained Xpath definition**
  - Must use locationPath or a union of location paths
  - Must use self, child, descendant, attribute and namespace axes
  - Cannot use context size and context position

  E.g. suppose the XPath 2.0 Filter contains

  ```
  //soap:Body//ns1:Echo[@a='23']
  ```

  to convert it we need to reverse it

  ```
  ancestor-or-self::ns1:Echo[@a='23']/ancestor-or-self::soap:Body
  ```
- This constraints still allow all the examples in the previous slide

Problem 3. Subsequent transforms don't see the entire document

- In the proposed processing model, each transform only sees the output of the previous transform
  - So suppose the original document is

```
<A>
  <B>
    <C/>
  </B>
</A>
```

And the there are two Xpath Filter transforms, the first getting the entire document, but the second one only getting a smaller subset e.g. if B is filtered out, then the second Xpath filter sees

```
<A>
   <C/>
</A>
```

So the second XPath filter wrongly assumes the C's parent is A, whereas it is actually B.

Problem 3. Contd

- **Constraint: To solve this we limit XPath Filter transform to be first transform, or to not use parent axis. (ancestor axis is ok)**

- Note: A similar problem also happens in XML namespaces and attributes in the xml namespace, i.e. B in the previous example could define a namespace or xml:base attribute, which is getting filtered out.
  - However we could solve this problem by “pushing down” all such filtered namespaces and xml attributes

Summary

- Proposed a “xml event” stream based processing model
- Only applicable for a subset, so we need to profile the XML signature spec for this.


ORACLE®