Comparing the Performance of Abstract Syntax Notation One (ASN.1) vs eXtensible Markup Language (XML)

Darren P Mundy
IS Institute
University of Salford
d.mundy@salford.ac.uk

David Chadwick
IS Institute
University of Salford
d.w.chadwick@salford.ac.uk

Andrew Smith
Salford Software
Manchester
Andrew.Smith@salfordsoftware.co.uk

Abstract

We describe a series of tests that have been carried out to measure the performance differences between eXtensible Markup Language (XML) and Abstract Syntax Notation One (ASN.1) with its Basic Encoding Rules (BER), for signed and unsigned attribute certificates. Our findings demonstrate that, as might be expected, XML encoded attribute certificates are approximately an order of magnitude greater in size than ASN.1 BER ones. The XML encoding process is initially faster than ASN.1 BER for small unsigned data structures, but the time taken increases more rapidly so that for large complex data structures ASN.1 BER is quicker than XML. However, ASN.1 BER decoding of unsigned data is always two to three times faster than XML decoding, regardless of data size. When digital signing is taken into account, ASN.1 BER is faster than XML for all data structures, and its performance advantage increases with data complexity. When validating signed data structures, ASN.1 starts at over 3 times faster than XML for simple attribute certificates, rising to an order of magnitude faster for complex attribute certificates. When transmission time is taken into account ASN.1 BER outperforms XML in all of our scenarios. For small comparable unsigned data objects across broadband connections, the performance difference is a factor of 5 greater, and for large signed data objects across 64k lines the difference is approximately an order of magnitude better. We conclude that where performance is required to be optimal there are better choices of data formatting and digital signing than XML, and one of those choices is ASN.1 with its BER.

Keywords: ASN.1, XML, Performance Testing, Digitally Signed Data, Attribute Certificates

1 Introduction

Over the past couple of years XML [1] has become the preferred syntax for the transfer of business information across the Internet. XML has received the wide spread backing and endorsement from major IT industry organisations like Sun Microsystems, IBM and Microsoft. The UK government is also firmly committed to making XML the basis for electronic government transactions through their e-Government Interoperability Framework (e-GIF) [2], e-Government Metadata Framework (e-GMF) [3] and the UK GovTalk program. XML has been touted as a true e-business enabler, but how does it compare against the transmission syntaxes it seeks to replace? There is a requirement to ensure that the propaganda is well placed. There is rarely a single technology that suits all situations in the IT industry. XML is an uncompressed textual syntax that remains in human readable form from creation to deletion. Therefore one must question whether it is really the ideal syntax to use in situations that require high performance.

In this paper we examine the comparison between XML and ASN.1 [4], a protocol specification language first standardised in 1984, and its Basic Encoding Rules (BER) [5].

We have found from past project experience [6, 7] that in the Internet world one of the critical success factors of user satisfaction is performance speed. Consequently this paper compares the creation, transmission and retrieval performances of ASN.1 and XML messages. It does not compare other factors such as the ease of use in building applications, or ease of debugging protocol errors etc. which are the sorts of factors that application programmers are interested in. These crucially are of little concern to the end users of the system. The key to many IT project failures has been the inability to perceive the needs of the end users [9, 15], and performance is one of them.

Our performance comparisons have been prompted by the impending adoption of XML as the protocol for electronic government services, for example the adoption of XML for the transmission of digitally signed electronic prescriptions.

2 ASN.1 and XML

ASN.1 is designed to describe the structure and syntax of transmitted information content. ASN.1 provides for the definition of the abstract syntax of a data element (or data type). The abstract syntax describes the syntactical structure and typed contents of data that are subsequently to be transmitted across some medium. The language is based firmly on the principles of type and value, with a type being a (non-empty) set of values. The type defines what values can subsequently be sent at runtime, and the value is what is actually conveyed across the medium at runtime. For example:

AllowedAccess ::= BOOLEAN

is the abstract syntax for a data element of type **AllowedAccess**, whose values at runtime can be one of the **BOOLEAN** values **TRUE** or **FALSE**. and **TRUE** might actually be a value conveyed within a binary encoded data stream at runtime (a possible binary encoding under BER will be, in hexadecimal notation, 01 01 FF, the first octet signifying type BOOLEAN, the second its length, and the third its value).

ASN.1 values are encoded before transmission using one of a number of different encoding mechanisms such as the Basic Encoding Rules (BER), the Distinguished Encoding Rules (DER), the Packed Encoding Rules (PER) or the recently introduced XML Encoding Rules (XER). The encoding rules specify how the values of the abstract data types are converted into byte strings ready for transfer. The recipient must usually be aware of the type definition before receipt, as this is not transferred but must be inferred from the context in which the message exchange takes place. The Basic Encoding Rules are very efficient and create Type, Length, Value (TLV) byte streams, so that the recipient, upon reading the length field, knows how many data bytes the value comprises. PER is even more efficient than BER, and is not based on TLV streams, so even greater optimisation can result. For example, PER never encodes the length of the value, unless it has to. If something has a fixed length, then the length field is not encoded.

During the transmission the ASN.1 data stream is never in a form readable by human operators (except when XER is used). Only when it has been transformed into some local data display format, prior to encoding or after decoding, can it be easily read by humans.

XML is a set of rules that allows data values to be encoded in text format. XML is a subset of the Standard Generalized Markup Language (SGML), but is also infinitely extensible. XML documents contain the information for transmission and consist of markup (which corresponds roughly to the "tag" and "length" parts in BER TLV encoding) and character data (which corresponds roughly to the "value" part in BER TLV encoding). Constraints can be imposed on the XML document structure with the provision of Document Type Definitions (DTD's) or XML Schemas. These describe the allowed markups that a conformant XML document can contain. Therefore in a DTD we might have a definition such as:

<!ELEMENT allowedAccess (#PCDATA)>

which states that the element **allowedAccess** is of type parsed character data (**PCDATA**) and can be processed by an XML parser on receipt.

<allowedAccess>TRUE</allowedAccess> might be a value of **allowedAccess** sent in an XML document to a recipient.

One can see immediately that XML is very verbose, and consequently creates large data streams (in this case 35 bytes compared to the 3 bytes of ASN.1 BER). XML is transferred in textual format with no binary encodings or compression. Furthermore, the recipient has to examine every byte received in order to determine the end of a data value. However, XML goes through no transformations and remains in a constant human readable format throughout the process.

In some sense we can say that DTD's / schemas map to the abstract syntax type definitions within ASN.1 and the XML documents map to the ASN.1 encoded byte streams. There are a few major differences between ASN.1 and XML/DTDs, with XML/DTDs lacking any concept of data type and ASN.1 being rich in built-in data types and supporting user-defined data types. Also, XML is very verbose, unlike ASN.1 encoding rules (except XER) that have been designed for optimal performance rather than human readability. However, from an application programmer's perspective, XML is easier to debug since the data stream can be read without any special software tools. Trying to read an ASN.1 BER or PER byte stream is very complex, but a number of free tools do exist to display ASN.1 data in its original source form e.g. dumpasn1 [16]. In addition, the XML 1.0 specification is a lot newer, simpler and easier to understand than the ASN.1 documentation, which has gone through several iterations and therefore contains many more sophisticated features.

3 Technology Used

Most of the software used for this comparison has been written in Java and has been compiled then executed using the IBM Java Development Kit (jdk) 1.3. This Java Virtual Machine (JVM) has been chosen due to its improved performance [8] over the Sun JVM. The default JVM that comes with IBM jdk 1.3 has been used and the operating system was Redhat Linux 6.2.

Some platform specific code has been written using the C language. Using standard Java it is not possible to obtain measurements of CPU utilisation and memory usage. It is possible to obtain the amount of free system memory and the amount of memory available to the JVM but we feel there are better ways of doing this than the default methods provided with a standard jdk. Code has been written in C, which can obtain the following information about Java thread performance:

User mode CPU utilisation
System mode CPU utilisation
Total number of pages in memory
Number of minor and major page faults

CPU utilisation is the time that the CPU is used by an application, with user mode utilisation being the time taken by a program running in a user state (e.g. in a loop or local function call) and system mode being the time taken by a low level routine such as opening a file or starting a process. A page in memory is a fixed number of bytes recognised by the operating system. A page fault occurs when a page is not in the buffer cache. Minor page faults are those that are satisfied in memory and major page faults are those that require disk access.

The C code has been written in such a way that it is possible to call the functions from within a Java program and therefore obtain system measurements from within Java that it would otherwise not be possible to make. The code used for testing is therefore platform dependant and will need to be modified to run under alternative operating systems.

The performance testing code has been written to execute using Linux as the OS, within which a process can easily be identified and measurements taken of it. The code used to obtain these measurements uses a publicly available library called libgtop that provides some of the functions required to obtain information about a process or the system as a whole.

Attribute certificates [10] are to be used as the transmitted data for protocol syntax comparisons. We have chosen attribute certificates for a number of reasons. Firstly they are very similar to public key certificates, which are a fundamental component of Web security. But unlike public key certificates where the payload is a public key, the content of attribute certificates (i.e. attributes) can be anything the designer wants them to be. Secondly, we are involved in generating a trial electronic prescription system which uses attribute certificate for the storage and retrieval of prescriptions. Health service professionals, especially those involved in busy pharmacy operations, require optimal performance in prescription processing. The UK Department of Health (DOH) has requested the use of XML for the information interchange and we would like to ensure through this comparison that XML is well suited for the requirements of prescription processing. The DOH has issued a number of DTD's [11] describing the expected structure of all electronic prescriptions. We have chosen to use the prescription structure specified in the EtpPrescribe DTD as the semi-complex attribute in our transmitted attribute certificates.

There is presently no definition for an attribute certificate in XML and there is equally no definition of the DOH prescription structures in ASN.1. Therefore we have generated these structures [12] using our knowledge of ASN.1 and XML and taking into account the existing XML definitions for public key certificates and signatures [13]. We have used BER encoding within our application to generate the encoded ASN.1 certificates. We appreciate that PER encoding creates an even greater optimised data stream but could find no zero cost implementation of PER in Java for our experiments. Thus we expect that our performance results for ASN.1 will not be the best ones possible, and could have been improved with alternate encoding rules such as PER.

4 Application For The Comparison

Our performance evaluation suite consists of client side (sender) and server side (recipient) applications. The attribute certificate is created by the sender and then transmitted to the recipient using standard sockets. The recipient firstly verifies the certificate and then parses it into a data structure for easy access to any of its data elements. The creation and subsequent retrieval of an attribute certificate involves the secure operations of electronic signing and signature verification respectively. These operations are required for electronic prescriptions and various other e-business processes e.g. digitally signed transactions and SSL.

However, not all e-business operations require digital signatures, so we decided to compare ASN.1 against XML with and without the overhead of the signing operation.

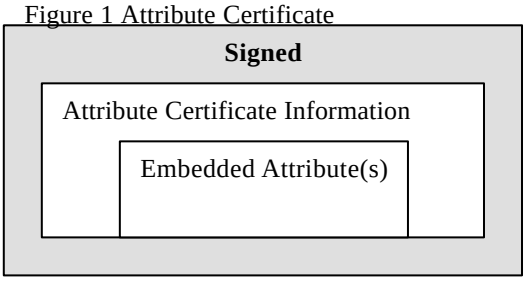
The Java software is capable of producing attribute certificates of any complexity. We chose to use a very complex attribute (a locally designed auditCertificate structure), a semi-complex attribute (a DOH EtpPrescribe structure) and a simple attribute (containing a simple Boolean attribute value). The signatures are produced for ASN.1 certificates using standard Java security classes in conjunction with our local Entrust Public Key Infrastructure (PKI). The private key used for signing is obtained from the PKI infrastructure. The signature for XML certificates is produced using a tool that comes with the XML Security Suite (XSS) available from IBM [14].

4.1 The Sender

The Java code used by the sender to generate the Attribute Certificates is capable of the following operations.

?? Production of signed or unsigned attribute certificates in ASN.1 with the help of a tool known as Snacc for Java, available from IBM.
?? Production of signed or unsigned attribute certificates in XML using the Java XML API (JAXP) available from Sun.
?? Zipped transmission of the data streams

Figure 1 Attribute Certificate



An attribute certificate, see Figure 1, consists of a structure called attribute certificate information, see Figure 2, which contains details about the issuer, the recipient, the validity time etc. as well as the embedded attribute(s). The Boolean, auditCertificate or EtpPrescribe structure is stored as an attribute within attribute certificate information. This attribute certificate information structure is then (optionally) signed and the signature method used and value of the signature are all amalgamated together to form the attribute certificate.

To generate this structure in ASN.1 we create the attribute certificate information then we use BER to encode it. An electronic signature is then generated from this and the unencoded attribute certificate information, the signature type and the signature value previously generated are all placed in a generated attribute certificate structure and then encoded using BER.

The XSS signature tool for XML certificates is passed an attribute certificate information object in XML, it then

Figure 2 Attribute Certificate Info Structure

```
Attribute Certificate Info ::= SEQUENCE {
version                 AttCertVersion,
holder                  Holder,
issuer                  Issuer,
signature               AlgorithmIdentifier,
attrCertValidityPeriod  AttCertValidityPeriod,
attributes              SEQUENCE of Attribute,
issuerUniqueID          UniqueIdentifier OPTIONAL,
extensions              Extensions OPTIONAL
}
```

canonicalizes the received XML and generates a signed XML document. The canonicalization process transforms the XML against a set of rules, which ensures that any syntactical differences between equivalent XML documents does not result in different binary representations. This is a necessary requirement for XML signature operations, otherwise signature validation is likely to fail. The public key can be included as part of the signed XML but because we do not include the public key in the ASN.1 structure we also removed public key transmission from the XML structure to ensure parity.

4.2 The Recipient

The recipient application is capable of the following for both XML and ASN.1:

?? Receiving the data as transmitted from the client.
?? Verification of the message (signature verification).
?? Parsing the received data into a suitable data structure.

The recipient has one main task, namely, to verify the received ASN.1 or XML data structure (the attribute certificate). The signatures are verified using the public key of the sender as obtainable from their Entrust profile. When the signature is verified the data is parsed into a data structure to enable easy access to any data member. With the XML message, the structure is parsed into a Document Object Model (DOM) structure, whereby the whole XML structure is created in memory. The ASN.1 is parsed into the Java class structure that can be used for both encoding and decoding ASN.1 attribute certificate structures.

Note that for XML, if a specific data item is required, then parsing using the Simple API for XML (SAX) parser is beneficial. With SAX parsing an event can be triggered when a specific field is found. This removes the need to parse the whole structure into memory, but is limited to the retrieval of a single field. ASN.1 does not have an equivalent to SAX parsing, and so no performance comparisons of SAX parsing vs. ASN.1 were made.

Verification of ASN.1 requires the BER encoded attribute certificate information value to be retrieved from the attribute certificate structure and compared against the supplied signature value (using the public key of the signer to decrypt it).

With XML the received structure is parsed for the signature value. Then the XSS tool is used to verify the signature against the public key and the XML content.

5 Performance Measurement

The performance comparisons took place on the same machine, see figure 3, to ensure any inaccuracies due to operating system specifics or hardware differences were not introduced.

Figure 3 Test Systems Specification

System	
?? CPU:	P3 650MHz
?? Memory:	256Mbytes
Software:	
?? OS:	Linux (Redhat 6.2)
?? Java:	IBM Jdk 1.3

Indeed to retain accuracy and credibility of the results it is essential that all external factors to the testing are kept constant. We did consider performing the encoding and decoding measurements on separate machines connected via a network, in client-server configuration, but because of the differences between the machines these measurement were withdrawn.

The only benefit these measurements would have provided us with were network performance, but we felt that these were not that relevant to this study. The sizes of the generated data streams should however be considered as part of the performance comparison, since the network transmission times will in general be directly proportional to the data sizes.

For each of the sender attribute certificate generation operations and the recipient verification operations the following measurements were taken:

?? CPU ticks for attribute certificate construction and verification

?? Process memory use for structure construction, (total number of pages in memory and difference over construction/verification period)
?? Number of page faults (minor and major) for structure construction and verification
?? The size in bytes of the completed certificates
?? The size in bytes of the zipped certificates
?? Elapsed time for construction and verification

Java programs execute within a virtual machine. In Java it is difficult to get access to individual threads for measurement purposes, so we have measured individual methods or activities, for example, different processes within the JVM for our time measurements and the actual JVM itself for CPU measurements, memory allocation etc. Measurements were made both before and after the execution of the method on the virtual machine and the difference taken. This gave a good indication of the resources used to perform the method.

The following tests were carried out and analysed:

Sender Timing of
?? ASN.1 attribute certificate generation using simple, semi-complex, and complex ASN.1 attributes (signed and unsigned)
?? DOM generated XML attribute certificate generation using simple, semi-complex and complex XML attributes (signed and unsigned)
?? Compression of all the XML generated certificates

Recipient Timing of
?? ASN.1 attribute certificate verification using simple, semi-complex and complex ASN.1 attributes (signed and unsigned)
?? DOM generated XML attribute certificate verification using simple, semi-complex and complex attributes (signed and unsigned)
?? Decompression of all XML generated certificates

Each of the tests was repeated one hundred times to allow for statistical variations in the results. We used a variety of attribute sizes to see if and by how much performance degrades as the attribute certificates get larger. The size of the attribute should affect every aspect of performance. We also thought it would be useful to look at the affect of compressing the XML attribute certificate prior to transfer to see if it would make a significant difference to XML transmission speeds, and to the overall performance of the XML processing.

6 Results

Overall we carried out twenty sets of client/server performance tests and these provided us with results from which to evaluate the performance of ASN.1 BER and XML.

6.1 Size Comparisons

The first factor to look at and perhaps the most limiting factor of the performance of XML is the relative size of XML output compared to ASN.1 output (see table 1).

	ASN.1 Unsigned	**DOM XML Unsigned**	**Zipped XML Unsigned**	**ASN.1 Signed**	**DOM XML Signed**	**Zipped XML Signed**
Simple Attribute	235 bytes	2880 bytes	710 bytes	384 bytes	3704 bytes	913 bytes
Semi-Complex Attribute	944 bytes	6210 bytes	1532 bytes	1060 bytes	7043 bytes	1737 bytes
Complex Attribute	1351 bytes	18297 bytes	4514 bytes	1483 bytes	19184 bytes	4733 bytes

Table 1 Size Comparisons (bytes)

From the results in Table 1, one can see that XML creates data blocks approximately an order of magnitude greater than BER encoded ASN.1. If the XML data is then compressed, it is still up to 3 times larger than the equivalent ASN.1.

6.2 Transmission Times

Throughout the testing stage we decided not to test transmission times between two machines. This decision was made simply because one cannot rely on a link speed being constant and we wished to remove any variable environmental factors from our testing. However, transmission time is a crucial factor in many real time applications and for an electronic prescribing system the amount of time it would take to receive a prescription at a pharmacy is crucial for normal operations.

	ASN.1 Unsigned	**DOM XML Unsigned**	**ASN.1 Signed**	**DOM XML Signed**
Simple Attribute	29 / 7	352 / 88	47 / 12	452 / 113
Semi-Complex Attribute	115 / 29	758 / 190	129 / 32	860 / 215
Complex Attribute	165 / 41	2234 / 558	181 / 45	2342 / 585

Table 2. Theoretical Times with Transmission over a 64kbps / 256 kbps link (ms)

Therefore if we reintroduce the concept of transmission times at this stage using theoretical constant link speeds of 64kbits and 256kbits assuming no lost packets – the sort of speeds one might optimistically achieve in a pharmacist's shop using the latest modem technology or broadband connection – we get the results shown in Table 2. In reality the performance over an actual modem link would be worse than the times theoretically calculated here. Protocol limitations on packet size will also cause performance degradation, e.g. the maximum segment size (MSS) in TCP/IP on an Ethernet network is generally set to 1460 bytes. Each of the ASN.1 data sizes in table 1 could fit into a

	Unsigned XML Zip time	**Unsigned XML Unzip time**	**Zipped Unsigned with Transmission Time**	**Signed XML Zip time**	**Signed XML Unzip time**	**Zipped Signed with Transmission Time**
Simple Attribute	40	10	136.66 / 71.67	40	10	161.45 / 77.86
Semi-Complex Attribute	45	15	237.01 / 96.75	45	15	262.03 / 103.01
Complex Attribute	45	15	601.02 / 187.76	45	15	627.76 / 194.43

Table 3. Theoretical Transmission Times over a 64 kbps / 256 kbps link (ms)

single TCP/IP segment. However, all of the XML data sizes are greater than the MSS, which would therefore result in longer transmission times because the data would need to be split across several packets. In fact if we take a comparison of ASN.1 against readable unzipped XML the performance deficit of XML is 9.6 times that of the simple ASN.1 signed attribute rising to 13.5 times for our most complex unsigned attribute, assuming a 64kbps link. So data size and corresponding transmission times are potentially a major factor limiting the performance of XML against ASN.1.

Perhaps compressing the XML prior to transfer will improve its overall performance. The compression was carried out via java classes at a maximum compression rate. The time taken to zip and unzip the XML was taken using Java calls to get the process time elapsed in milliseconds. If XML is zipped prior to transfer, then the advantage of readability during transfer disappears, but the performance can significantly increase. Table 3 shows that after performing a zip operation on XML, the XML transaction is between 3.5 (for complex unsigned attributes) and 10 times (for simple signed attributes) as big/slow as ASN.1.

6.3 Encoding/Decoding of Unsigned Data

Turning now to the encoding and decoding of ASN.1 and XML data structures. Table 4 shows the times taken by the sender to encode a simple attribute, a semi-complex attribute and a complex

attribute, without signing them. Table 5 shows the corresponding times for the recipient to decode the byte streams.

For the smaller unsigned attributes, sender performance of ASN.1 is significantly worse than XML but is better for the complex unsigned attribute. In contrast, recipient performance of XML is always significantly worse than ASN.1. This can be explained as follows. The time for DOM encoding is directly proportional to the size of the generated XML, and requires virtually no start up time. However the encoding time of one data element appears to be relatively slow.

	ASN.1	**DOM XML**	**ASN.1 Performance Comparison**
Simple Attribute	6.83	2.66	-60%
Semi-Complex Attribute	8.98	4.46	-50%
Complex Attribute	10.54	14.88	+40%

Table 4. Sender Encoding Times of Unsigned Data (ms)

	ASN.1	**DOM XML**	**ASN.1 Performance Comparison**
Simple Attribute	1.63	4.46	+170%
Semi-Complex Attribute	2.49	5.5	+120%
Complex Attribute	3.52	9.07	+157%

Table 5. Recipient Decoding Times of Unsigned Data (ms)

ASN.1 on the other hand has a relatively large initial start up time, but the encoding, once started, is relatively fast. As the XML becomes larger its initial performance advantage is eaten away until it starts to lag behind ASN.1. On the recipient side, ASN.1 has both a faster start up time and a faster decoding time, so always outperforms XML.

6.4 Encoding/Decoding of Signed Data

For secure transmissions we require signature operations to be performed on the ASN.1 and XML data structures. From our results it appears that ASN.1 signing and verification far outperforms XML signing and verification. When signature operations are required sender ASN.1 encoding outperforms XML encoding for simple attributes by 20% rising to 80% for complex attributes, see table 6. For recipient decoding and signature verification the difference is even more pronounced, with ASN.1 outperforming XML by 450% for a simple Attribute and close to a 1000% for the largest attribute we tested, see table 7.

It should be noted that the XML signature process is not standard yet so it is possible that the signing software may well be immature and not optimised. Consequently there may be an opportunity for the XML signing and verification process speeds to be improved. However, to counteract that we know that ASN.1 PER outperforms ASN.1 BER, and, more importantly, there is more processing involved in generating XML signature documents than ASN.1 ones, and this will not change. So it may not be possible for optimised XML to significantly close the gap on optimised ASN.1. Of considerable interest is the fact that the increase in ASN.1 timings is minimal with increasing complexity of the signed data structure, whereas the increase in the XML timings is very significant, see Table 8 and graphs A and B.

For secure transmissions we require signature operations to be performed on the ASN.1 and XML data structures. From our results it appears that ASN.1 signing and verification far outperforms XML signing and verification. When signature operations are required sender ASN.1 encoding outperforms XML encoding for simple attributes by 20% rising to 80% for complex attributes, see table 6. For recipient decoding and signature verification the difference is even more pronounced, with ASN.1

outperforming XML by 450% for a simple Attribute and close to a 1000% for the largest attribute we tested, see table 7.

	ASN.1	DOM XML	ASN.1 Percentage Comparison
Simple Attribute	94.82	113.36	+20%
Semi-Complex Attribute	100.28	125.85	+26%
Complex Attribute	102.79	184.12	+80%

Table 6. Sender Encoding Times of Signed Data (ms)

	ASN.1	DOM XML	ASN.1 Percentage Comparison
Simple Attribute	5.92	26.62	+350%
Semi-Complex Attribute	6.01	38.96	+550%
Complex Attribute	6.16	67.22	+1000%

Table 7. Recipient Decoding Times of Signed Data (ms)

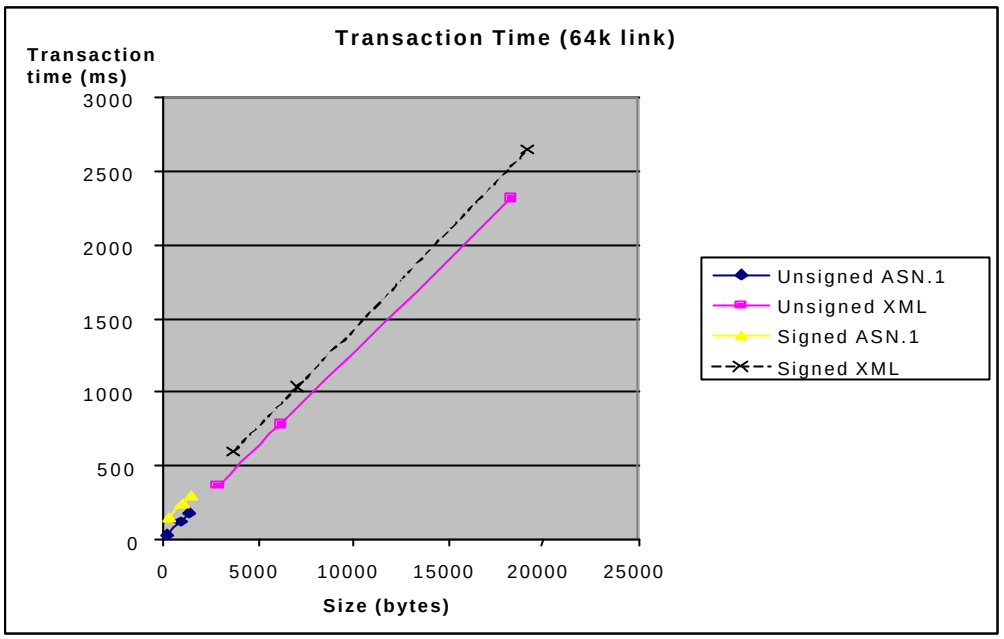
It should be noted that the XML signature process is not standard yet so it is possible that the signing software may well be immature and not optimised. Consequently there may be an opportunity for the XML signing and verification process speeds to be improved. However, to counteract that we know that ASN.1 PER outperforms ASN.1 BER, and, more importantly, there is more processing involved in generating XML signature documents than ASN.1 ones, and this will not change. So it may not be possible for optimised XML to significantly close the gap on optimised ASN.1. Of considerable interest is the fact that the increase in ASN.1 timings is minimal with increasing complexity of the signed data structure, whereas the increase in the XML timings is very significant, see Table 8 and graphs A and B.

	ASN.1 Recipient Performance	ASN.1 Sender Performance	XML Recipient Performance	XML Sender Performance
Simple Attribute	100%	100%	100%	100%
Semi-Complex Attribute	101.52%	105.75%	146.35%	111.01%
Complex Attribute	104.05%	108.40%	252.52%	162.42%

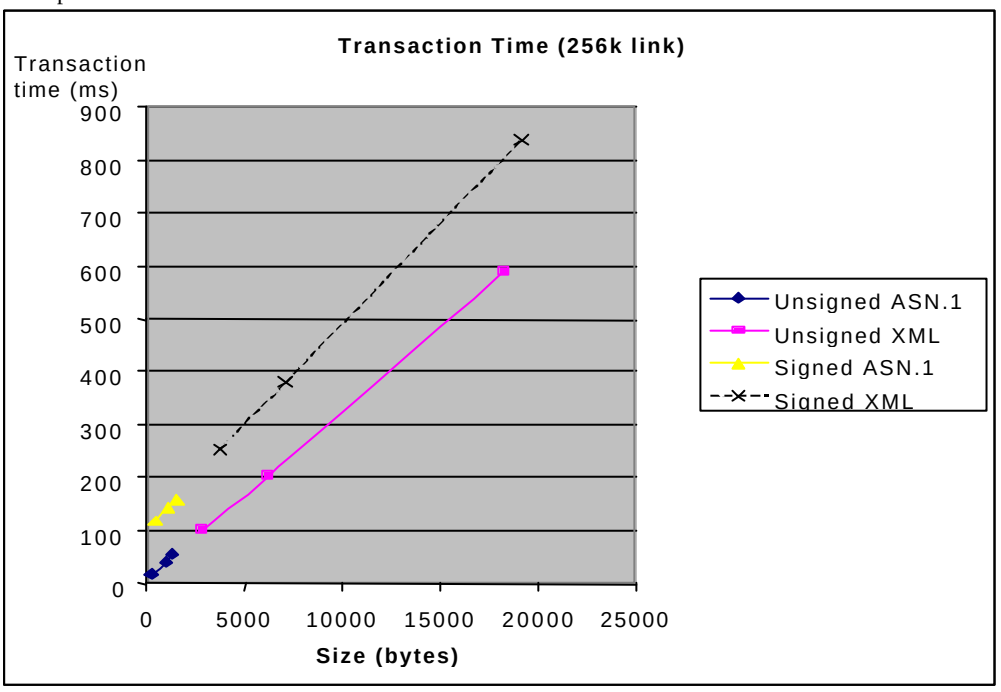
Table 8. Average Performance Decrease (%) with Complexity of Signed Attributes

When the theoretical transmission time over a 64k link is combined with the encoding and decoding times there really is no contest between ASN.1 and XML. Unsigned and signed ASN.1 always outperforms XML by approximately an order of magnitude, see Graph A. Even with broadband access and assuming the maximum throughput of 256kbps, the performance deficit of XML in the most favourable scenario (unsigned simple attribute) is still 5 times worse than ASN.1 (see Graph B). With current technologies typically in use today in pharmacist's shops and doctor's surgeries, we think that our figures are realistic.

Graph A



Graph B



6.5 System Timings

Many of the system measurements resulted in significant variations between program runs. This can be explained in terms of cache usage, background operating system tasks (in a multitasking environment) and slight variations in the code. For a number of the system tests the differences in the attributes didn't have a large affect on the measurements, consequently in a number of cases the system variations were enough to swamp the expected variances between attribute sizes. For all the following test results a +/- factor is provided to show the amount of system measurement variation.

	ASN.1 (Sender) (+/- 0.008)	DOM XML (Sender) (+/- 0.008)	ASN.1 (Recipient) (+/- 0.008)	DOM XML (Recipient) (+/- 0.015)
Simple Attribute	0.013 / 0.17	0.007 / 0.25	0.009 / 0.024	0.063 / 0.15
Semi-Complex Attribute	0.019 / 0.18	0.008 / 0.28	0.009 / 0.027	0.064 / 0.17
Complex Attribute	0.023 / 0.18	0.022 / 0.37	0.005 / 0.030	0.067 / 0.20

Table 9. Average Sender and Recipient CPU User Time for Unsigned Data / Signed Data (in ? s)

	ASN.1 (Sender) (+/- 0.0003)	DOM XML (Sender) (+/- 0.0003)	ASN.1 (Recipient) (+/- 0.0005)	DOM XML (Recipient) (+/- 0.0015)
Simple Attribute	0.0002 / 0.0006	0.0003 / 0.0030	0.0005 / 0.0005	0.0012 / 0.0016
Semi-Complex Attribute	0.0003 / 0.0006	0.0005 / 0.0030	0.0006 / 0.0006	0.0014 / 0.0022
Complex Attribute	0.0006 / 0.0010	0.0006 / 0.0040	0.0008 / 0.0009	0.0015 / 0.0028

Table 10. Average Sender and Recipient CPU System Time for Unsigned / Signed Data (in ? s)

6.6 CPU User Time and System Time

We measured both CPU user time and CPU system time for sender and recipient operations to ascertain just how much of a burden ASN.1 and XML were on the actual processor itself. If we first consider the unsigned attribute certificates we find that the sender of XML requires a lower amount of CPU user time than the sender of ASN.1. However, the ASN.1 recipient always outperforms the XML recipient. When dealing with signed data ASN.1 requires lower CPU user time than XML for both sender and recipient for all attribute complexities.

Generally applications require lower amounts of CPU system time than user time and our test suite was no different with the values being barely over zero, as shown in table 10. The only significant point to note is that the system time required by XML in almost every case was more than the system time required for ASN.1.

6.7 Memory Allocation and Page Faults

Dynamic memory allocation was measured on both the client and recipient sides. The average amount of dynamic memory in kb allocated over the tests performed was then calculated. This memory is in addition to the memory allocated on initiation to the JVM of a default 1Mb. The results show that without the overhead of security XML required lower amounts of dynamic memory allocation than ASN.1. If we look at the construction method for each then perhaps we can theorise as to why. The ASN.1 requires a large number of class instantiations and ultimately destructions, whereas the XML

	ASN.1 (Sender) (+/- 81.92 / 81.92)	DOM XML (Sender) (+/- 81.92 / 573.44)	ASN.1 (Recipient) (+/- 81.92 / 81.92)	DOM XML (Recipient)(+/- 81.92 / 491.52)
Simple Attribute	450.56 / 409.6	40.96 / 614.4	122.88 / 737.28	122.88 / 942.08
Semi-Complex Attribute	450.56 / 409.6	81.92 / 614.4	122.88 / 778.24	122.88 / 942.08
Complex Attribute	450.56 / 409.6	81.92 / 614.4	122.88 / 778.24	163.84 / 1187.84

Table 11. Average Dynamic Memory Allocation for Unsigned / Signed Data (in Kb)

	ASN.1 (Sender) (+/- 1 / 3)	DOM XML (Sender) (+/- 2 / 3)	ASN.1 (Recipient) (+/- 1 / 3)	DOM XML (Recipient) (+/- 2 / 3)
Simple Attribute	4 / 6	0 / 9	5 / 7	4 / 12
Semi-Complex Attribute	4 / 6	2 / 9	5 / 8	4 / 12
Complex Attribute	4 / 9	2 / 10	5 / 8	6 / 14

Table 12. Minor Page Faults per 100 operations for Unsigned / Signed Data

application uses fewer classes and therefore has lower initial memory requirements. It is also possible that the problem could be with the ASN.1 compiler used. However, when signatures are required, the XML application has to do much more work than the ASN.1 application e.g. canonicalization is required, resulting in the appropriate increase in memory requirements.

Within our performance tests we only ever encountered a major page fault once. Minor page faults were minimal and the value for all tests never rose above 14 in every 100 tests. This is a very low figure. The number of page faults affects execution time and if any of the test results had resulted in particularly high numbers of faults then we would have needed to consider the affect that these would have had on the process times. However, all of the tests results shown in table 12 show small numbers and minimal differences between the tests for ASN.1 and XML, except that signing always generated more faults than no signing.

7 Conclusions

In many environments XML is a preferred way of encoding business transactions, since the messages are readily viewable by web browsers. If these environments involve simple XML messages, without digital signatures, then XML performs adequately and the benefits of XML can be realised. In fact we have found that simple XML message creation is more efficient than creating an equivalent ASN.1 byte stream. For critical real time systems where digital signing of complex data structures is required, and where performance is a key success factor, such as an electronic prescribing system for example, we have shown that signed complex XML messages can be up to a 1000% slower to decode than an equivalent ASN.1 message.

XML is easy to manipulate and easy to understand, all factors which make it attractive to both senior management and developers. However, the key to many IT project failures has been the inability to perceive the needs of the end users, and performance is one of them. We believe that in a real time system dealing in multiple transactions a second and requiring strong authentication through digital signatures, XML formatting is not a good protocol to choose. Within an electronic prescription system for instance, the users who would take the biggest performance penalty would be the pharmacists receiving the digitally signed prescriptions, and these are precisely the users who require optimum performance when they are trying to dispense drugs rapidly in a busy pharmacy. The prescribing GPs who have to create and sign the prescriptions would also be penalised by using XML formatted prescriptions instead of ASN.1 formatted one. This might ultimately result in user dissatisfaction and perhaps even total system failure. Since end users are aware of system performance and not of the underlying data encoding mechanisms, we believe that performance figures are an important factor in system design. In this paper we have shown that with digitally signed messages ASN.1 can significantly outperform XML by over an order of magnitude.

Acknowledgements

This research was funded by the UK EPSRC under grant number GR/M83483 . The authors would also like to thank Entrust Technologies for making their PKI security software available to the university on preferential terms.

References

[1] “Extensible Markup Language (XML) 1.0 (Second Edition)”, W3C Recommendation, 6 October 2000. See http://www.w3.org/TR/REC-xml (Current May 2002)

[2] Cover, R. “e-Government Interoperability Framework (e-GIF)”, May 2002. See http://www.oasis-open.org/cover/egif-UK.html (Current May 2002)
[3] “The e-Government Metadata Framework (e-GMF)”. See http://www.e-envoy.gov.uk/publications/frameworks/metadata/metadata.htm (Current May 2002)
[4] ITU-T, X.680 : ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1:1998, Information Technology - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation
[5] ITU-T, X.690 : ITU-T Recommendation X.690 (1997) Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
[6] Chadwick, D.W., New, J.P., McDowell, D.M. and Mundy, D.P. 2001, ‘Providing Secure Access to Confidential Patient Information Detailing Diabetic Condition’, *Proceedings of the International Conference on Internet Computing 2001 Volume I*, Editors P.Graham, M.Maheswaran, R. Eskicioglu, CSREA Press, pp 535-541.
[7] Chadwick, D.W., Carroll, C., Harvey, S., New, J., Young,A. J. “Experiences of Using a Public Key Infrastructure to Access Patient Confidential Data over the Internet”, Proceedings of the 35th Annual Hawaii International Conference on System Sciences 2002 (HICCS 2002) Ed. Ralph H. Sprague, Jr. Abstract p 156. Also available from http://www.hicss.hawaii.edu/HICSS_35/HICSSpapers/PDFdocuments/HCTHC01.pdf (Current May 2002)
[8] T. Suganuma, T. Ogasawara, M. Takeuchi, T. Yasue, M. Kawahito, K. Ishizaki, H. Komatsu, and T. Nakatani. “Overview of the IBM Java Just-in-Time Compiler”, IBM Systems Journal, Volume 39, Number 1, 2000. See http://www.research.ibm.com/journal/sj/391/suganuma.html (Current May 2002)
[9] Beynon-Davis P. (1999), “Human error an information systems failure: the case of the London Ambulance service computer aided despatch system project”, Interacting with Computers 11, 699-720.
[10] ISO/ITU-T Rec. X.509 (2000) The Directory: Authentication Framework
[11] “Electronic Transmission of Prescriptions (ETP)” Department of Health. See http://www.doh.gov.uk/pharmacy/etp.htm (Current May 2002)
[12] ASN.1 and XML Specifications and examples, See http://sec.isi.salford.ac.uk/EPP/public/comp.html (Current May 2002)
[13] Eastlake, D., Reagle, J., Solo, D. “(Extensible Markup Language) XML-Signature Syntax and Processing” RFC 3275, March 2002
[14] “XML Security Suite (XSS)”, IBM Alphaworks Technology. See http://www.alphaworks.ibm.com/tech/xmlsecuritysuite (Current May 2002)
[15] Ewushi-Mensah, K. & Przasnyski, Z. (1994), "Factors contributing to the abandonment of information systems development projects", *Journal of Information Technology* 9, 185-201.
[16] dumpasn1 can be found at http://www.cs.auckland.ac.nz/~pgut001/ (October 2002)