XML :
A TECHNOLOGY PERSPECTIVE AND SECURITY IMPLICATIONS

GERARD ANG

UNIVERSITY OF WALES, ABERYSTWYTH

2003

XML :
A Technology Perspective and Security Implications

GERARD ANG

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in the University of Wales

Supervisor: Dr Fredrick W. LONG

University of Wales, Aberystwyth

15 September 2003

DECLARATIONS

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed: ________________________ (Gerard ANG)

Date: 15 September 2003

STATEMENT 1

This dissertation is being submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Signed: ________________________ (Gerard ANG)

Date: 15 September 2003

STATEMENT 2

I hereby give consent for my dissertation, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organizations.

Signed: ________________________ (Gerard ANG)

Date: 15 September 2003

Table of Contents

Chapter 1

XML History: The Internet and Markup Languages

The development of the Extensible Markup Language (XML) is closely linked to the evolution of the Internet and the World Wide Web. This chapter describes:

- *The brief history of the Internet and the Web Architecture*
- *Interpreting documents on the Web*
- *The concept of markup languages*
- *SGML and HTML : Structure, advantages and limitations*

1.0 The Internet

The Internet started as a project by the US Department of Defense in the 1960s. The aim was to develop a network of linked computers that could continue to communicate even if a part of the network were destroyed. Today, the Internet provides the infrastructure and associated protocols to support many different computer applications. It offers a platform for many services to be delivered electronically throughout the world.

Conceptually, the Internet is the infrastructure, while the World Wide Web is the application suite that uses this infrastructure. The Web enables information transfer and sharing. At the heart of the Internet is a suite of networking protocols known as the Transmission Control Protocol / Internet Protocol (TCP/IP).

While being developed, the challenge was to create a system that would be accessible to all the diverse computer systems in use. A number of potential solutions existed then, such as Postscript. However, these languages were found to be too complex and eventually, the Standard Generalized Markup (SGML) was selected as the basis for Web content. Later, SGML was used to define the Hypertext Markup Language (HTML) as a language for producing Web pages. These pages are designed to be for human consumption. That is, human-centric, as opposed to being machine-centric [31].

2.0 The Web Architecture

The Web architecture is based on a client-server model. The client application is the browser, and other software applications function as servers (e.g. Netscape Application Server, Apache).

The browser allows users to locate Web pages by using specific page addresses known as a Uniform Resource Locator (URL). The Hypertext Transfer Protocol (HTTP) is used to enable the client browser to communicate with the Web server (e.g. to locate, download, interpret, and display 'pages').

The browser sends a HTTP request to the server (the request is encapsulated by TCP/IP and carried across the network). The server then locates the requested page, and sends it to the browser. The browser sees the body, interprets and presents it [16].

3.0 Interpreting Documents on the Web

Although most browsers support basic HTML definitions, different vendors (e.g. Microsoft, Netscape) have added their own proprietary features and extensions (e.g. non-standard tags) to expand basic HTML and increase browser functionality. These have resulted in interoperability problems. Because the browser interprets the contents of a HTTP response according to its internal rules, the same web page might not look or behave in the same way when different browsers are used [13].

Furthermore, vendors may have added proprietary extensions that can only be accessed through proprietary scripting languages. A vendor might add platform-specific scripting capabilities to their own browser, allowing access to specific browser functionality not directly supported by HTML [6]. Ironically, by taking advantage of such scripting capabilities, a web developer might inevitably restrict the range of browsers that can correctly handle the document he is developing.

4.0 The Concept of Markup Languages

A markup language is a mechanism for identifying the structure in a document. It describes how the text within a document is structured according to tags. These tags are used to 'markup' words or sections, so as to indicate actions or identifications. Hence, a document can be 'marked up' to facilitate information search, or to relate information with other markup documents [3], [16].

Among other things, a markup language allows users to:

- add meaning to the underlying text
- provide structure to the document; and
- provide basic layout instructions.

For example, to bold a text, add two "BOLD" tags enclosed between two angle brackets '<' and '>', as in '<BOLD> **urgent<**/BOLD>'. Thus, a markup language defines a fixed set of tags that describe a fixed number of elements.

The browser interprets the markup information and acts on it. In some cases, if a browser encounters a tag that the browser cannot understand, it will treat it as a default text and displays its contents.

5.0 Standard Generalized Markup Language (SGML)

SGML is a markup language that describes the structure of a document. It is a complex and powerful language, designed to deal with large sets of data. Being a meta-language, SGML can be used to define other languages, such as HTML. Hence, HTML is one of SGML's Document Type Declaration (DTD) [31].

5.1 The Advantages of SGML

Some advantages of SGML are [19]:

1. It is intelligent and adaptable, and best used in managing large sets of data, such as dictionaries and encyclopedias.

2. It is a universal standard and is supported by many software vendors. Because of this universality, a document developed on the SGML standard is much more likely to survive the test of time, than one built around more ephemeral, vendor-specific standards.
3. It describes the data, and not just the way in which it is presented.

5.2 The Limitations of SGML

The weaknesses and limitations of SGML are [3]:

1. It is extremely general-purpose
2. It is complex. Its specifications comprise of around 500 pages
3. The tools are very expensive to acquire (e.g. Framemaker)
4. It lacks power in linking and portability. Hence, interoperability is very limited.

6.0 Hypertext Markup Language (HTML)

HTML is a language defined by the SGML. HTML was created to counter the complexity of SGML, while retaining some of SGML's power [8].

6.1 The Structure of HTML

A HTML document is structured as it follows the rules associated with the underlying standard. However, a HTML document need not have to strictly adhere to the standard [13].

HTML takes the form of a series of tags enclosed by angle brackets < >. HTML documents should begin with a <!doctype> statement. This statement identifies the version of HTML to which the document conforms. The statement references the external DTD that contains the SGML definitions. A user can use a validator program to check if the document conforms to the standards of the stated DTD. However, due to relaxed requirements of HTML, relatively few documents on the Web would pass extremely strict validation [29], [31].

6.2 HTML Document Presentation

A series of tags are used to define how a document should look like on the browser. For example, tags such as <h1> or <p> are used to specify the structure of a document and its logical hierarchies [13]. Or, tags can be used to specify how a page should look when displayed, which fonts should be used, and so on.

HTML is weak in final formatting. Workarounds typically have to be used to achieve the desired effects [3]. Furthermore, as mentioned earlier, because of browser differences, the effectiveness of using HTML to gain strong presentation control over web pages is hindered.

6.3 The Advantages of HTML

One of the main reasons the Web caught on quickly was because users need not be experienced programmers to develop web pages. HTML is "forgiving" of sloppy programming, or unplanned development.

The advantages of HTML include [8]:

1. It is easy to learn, as it is a simple page description language.
2. It is simple to use. The user employs a series of tags to define how a document should look like
3. It is forgiving in that strict adherence of syntax rules is not mandatory
4. HTML pages can be generated easily using a graphics editor. The obligatory chore of manually entering tags is not required. HTML authoring tools such as FrontPage generates the HTML tags for the user
5. It is a cross-platform solution
6. It is widely supported (that is until browsers decided to have their own functionalities)
7. It is a mature language. Users have worked around many of the earlier limitations
8. It is pervasive

6.4 The Limitations of HTML

Some limitations of HTML include [3], [13]:

1. It is not suitable for explicit queries about the page content. Although a HTML page contains a mix of content and presentation, HTML only describes the appearance of the page, that is, how a page should appear
2. It is version dependent. Certain browsers may not support a particular HTML version
3. It does not enforce best practices in programming techniques. For example, the "begin" and "end" tags are not mandatory
4. When implemented by a vendor, it is dependent on both client and server
5. It is not extensible
6. It does not allow individual elements to be marked up semantically

Although there are many limitations of HTML, its simplicity and straightforwardness ensures its widespread use.

However, it was only recently that users understood the importance of content classification. This is something that HTML does not provide.

7.0 After SGML and HTML, Comes XML

As explained earlier, while SGML is too complicated, HTML is not complicated enough to handle Web page requirements. Hence, HTML's inherent restrictions have led to the development of proprietary extensions by various vendors. This in turn has put a negative effect on the universality of HTML-formatted documents [18], [31].

To overcome HTML's limitations, since SGML is HTML's "mother tongue", an obvious approach would be to see if SGML can be improved. Developers concluded that a simplification of SGML is the answer, not a broader-based HTML [3]. Therefore, a new language allowing users to implement their own applications on the basis of a simple standard is required. Enter XML.

Chapter 2

Structured Information

One of the reasons for the paradigm shift towards business disintermediation and disintegration of operations is the availability of easily accessible information. Enabling technologies to process data and information has made this possible. This chapter describes:

- *The need for information exchange*
- *The concepts of data and information*
- *The concept of structured data and information*
- *The concept of information context and vocabulary*
- *The concept of data portability*
- *The benefits of Structured Data*
- *Processing Implications of Structured Data*

1.0 Introduction

The use of Information Technology in business operations has resulted in the disintermediation of operations that rely on easily accessible information flows. For example, in the insurance business, the services of a "middle-man" or intermediary to broker a deal is less needed now, as information flow between the customer and insurance company, and vice-versa, can be done electronically [31]. This ability to easily share and exchange information has been cited as a major competitive advantage for many companies.

2.0 The Need for Information Exchange

The amount of data and information made available has never been larger. With the trend towards globalization, the need for companies to exchange data and information in an effective and efficient way has never been greater. Some of the business drivers for effective information exchange are:

- Controlling and reducing operational costs
- Increasing sales and revenues
- Increasing competitive edge

In an operationally demanding setup, it is quite easy to appreciate that a smooth, efficient and effective operation of a company can only be effected if information exchanges are done effectively, efficiently and seamlessly.

Furthermore, the growth of the Internet has changed businesses' perception of how information should be accessed or exchanged [5]. In both internal and external transactions, information exchanges can be for person-to-person (humans centric), and application-to-application (machine centric).

3.0 Data and Information

- 1.52
- 360
- 12:45

Although we can infer some meaning for each data shown above, in its own right, each data does not make much sense. For example, the data "1.52" could mean $1.52, or 1.52 metres. Without additional context, the information is incomplete. Hence, to be able to exchange data, one must provide additional context that enables the meaning to be obtained.

In the case of information, two possible definitions for "information" could be [1]:

- Data that has been organized and presented in a systematic way, to clarify the underlying meaning; or
- Data that is arranged and organized, so that it has meaning

These two definitions cover the core aspects of 'structure' and 'organization'.

4.0 Structured Data and Information

Data and information can be in the following forms:

- In free-form
- In a semi-structured form (e.g. a newspaper article)
- Structured (e.g. phone directory)

When data is structured, it means that [13]:

- a vocabulary that defines the elements in the document can be specified
- the relations between these elements are specified

For example:

- in a customer contacts list, every contact must list a phone number and email address
- all fields in the database must be entered. E.g. every 'book' field must have an author

When data is structured, the data is transformed from a state of an unordered mass of free-form information, to a structured, searchable, and understandable store of data [31].

Consider a webpage advertisement of a holiday shown below:

Diving Holiday	Go diving in Wales For $390
5 nights in Aberyswth, including full dive equipment, based on twin sharing basis	Departing Heathrow June 21, 2002

Refer to the advertisement above, one could:

- easily relate this webpage information to a page from a printed advertisement brochure and treat the information in a single, complete item; or
- equate it in a data-centric way, i.e. putting the information into some form of structure. E.g. defined as a "Holiday Information",

An example of relating the information in a data-centric way is shown below:

Holiday Information

Holiday type:	Diving
Country:	Wales
Location:	Aberyswth
Number of nights:	5
Departure date:	June 21, 2002
Price:	$390
Price basis:	Twin sharing
Additional info:	Full dive equipment provided

From the above, we can see that a structure provides the context for information content. Such a structure or vocabulary will facilitate easier exchange of information. If one is to exchange information on anything other than on a random basis, the definition and agreement of underlying structural formats and vocabularies are critical [3].

5.0 Providing Context and Vocabularies

Different vocabularies would be needed to support different applications.

For example, from the "Holiday Information" examples shown earlier, the definition used to exchange "Holiday Information" differs from the definition used for, say, (a) currency exchange rates, and for (b) recipes, as shown below.

(a) **Currency Exchange Rates Vocabulary**

Base currency:	GBP
Target Currency:	USD
Date:	21/06/2002
Time:	12:34
Rate:	1.56

(b) **Recipe Vocabulary**

Name:	Shepherds Pie	
Portions:	12	
Ingredients:	Flour	2 cups
	Butter	1 cup
	Sugar	5 teaspoons
Method:	Preheat oven to 220^0 C, etc…	

6.0 Data Portability

The concept of using well-defined data structures is not new. For example, conventional databases provide context through data dictionaries and column definitions.

However, the portability of data has always been an issue. Until recently, data formats are typically proprietary (closed standard, as opposed to an open standard), even when the access syntax is portable (e.g. SQL) [31]. As the use of information technology broadens, the requirement for standardization (open standard) across platforms has become a major issue.

Programming languages like Java has enabled portable code for cross-platform capability. Therefore, having portable data is the next step for application-independence capability. This will enable the passing of data between applications without any loss of meaning [1].

The way in which the Web is used has changed greatly. In the past, it is seen more as an information repository, where data is largely unstructured [13]. Now, with the emergence of Web-based applications, the Web is used more as a vehicle for machine-to-machine information exchange (machine-centric), rather than only as a mechanism to deliver information to humans (human-centric) [1].

Hence, the Web has moved from being an environment of free form, unstructured, document repository, to a structured environment for application-processing [26]. In this machine-to-machine environment, having structured data is critical to ensuring data portability.

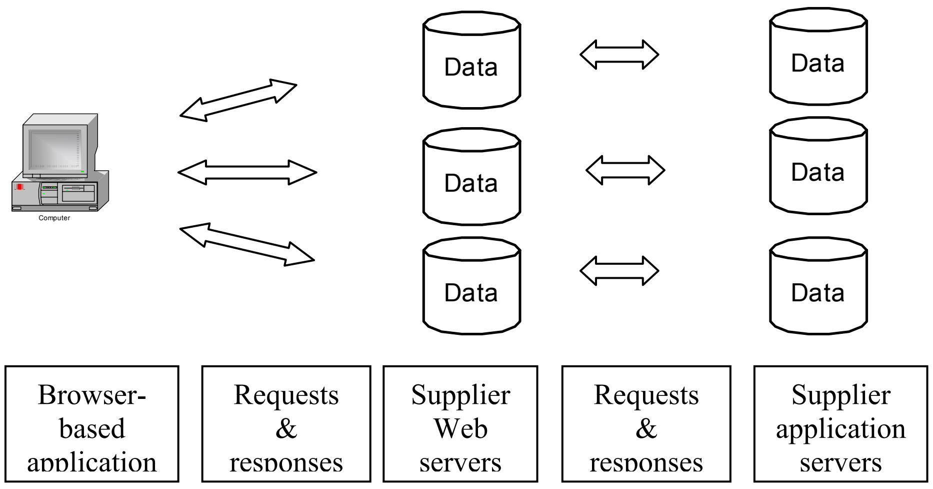
7.0 Benefits of Structured Data: Application Independence

One benefit of having common structures and open standards is that it enables low degrees of application coupling. If data from different sources adhere to commonly agreed structures, applications can be built to consolidate the data and present it in a unified way [31]. For example, if a food manufacturer requires more sugar, he would typically need to make many phone calls to attempt to source for sugar supplies.

However, if he had adopted a standard format for structured data that is in common with the rest of the food and sugar industry, the situation would have been different. For example:

- The manufacturer would have a browser-based application for locating suppliers. He does not have to refer to his old contact list of sugar suppliers or other directories, which might not be up-to-date in any case.
- The application automatically makes request for sugar from potential sugar suppliers, and receive responses in the standard industry format
- The application could then sort and display the information from potential suppliers, for example, by location, by volume of sugar available, and by price
- The application could then place the order with the supplier

The application could have the additional benefit of being able to work with any new suppliers that conforms to the common standard. This means that the food manufacturer will always have an up-to-date list of sugar suppliers.



Having structured information can be a key enabler to many other types of applications, such as data aggregation, search and retrieval, and media-independent publishing [3]. These applications could be document-centric, or data-centric, and some hybrids of both. For example:

- E-commerce applications
- Concise search and retrieval applications that makes use of known data formats. Able to tightly target the required information
- Producing documents which are independent of any particular publishing media. The user can apply the appropriate format processing later and target the document to any output media, like paper, etc.

8.0 Processing Implications of Structured Data

Web pages are commonly used as mechanisms to provide access to information from other systems. For data-centric applications, static web pages are populated by server scripts from data contained in a range of systems, such as databases and Enterprise Resource Planning applications [13].

At the same time, data can be accessed from a range of proprietary sources and then converted into standard formats. Thus, the data processing landscape can become quite complex [31]. As the use of structured data increases, it is expected that the use of static Web pages will decrease.

9.0 Conclusion

The Internet has evolved and transformed from being a publishing medium to being an application-processing environment.

Portability of data is now required. As businesses move entire systems to the Intranet, they will need to rely on structured documents to build complex structures of documents using custom tags. This cannot be achieved easily with HTML, as HTML was not designed to deal with structured data nor support the flexible interchange of structured information [3]. Hence, it cannot meet many of today's business needs for information exchange.

Alternative technologies are needed to meet the new requirements for information exchange, and the emerging application-processing environment. As it was no longer practical to stretch the limits of HTML, XML was developed to meet this new challenge.

Chapter 3

Extensible Markup Language (XML): An Overview

The XML specification evolved from the SGML and HTML. This chapter provides an overview of XML by describing XML's:

- *History and goals*
- *Document Structure (DTDs, Schemas, Document Validity and Well-Formness)*
- *Core Elements and Functionality*
- *Concepts of Extensibility and Metadata*
- *Outputting, Transforming and Re-purposing XML*
- *Key difference with HTML*
- *Standards, History and Landscape*

1.0 Introduction

XML is formal markup language that provides an open-standard format for describing documents containing structured data and information. It is a system used for defining, validating, and sharing document formats. 'Document' refers to structured text and data formats such database schemas.

XML enables the creation of customised markup languages for specific documents and domains. Using a methodology for tag creation, tags are used to distinguish document structures and attributes, to encode document information. Once defined, tags are mixed with data to form a XML document [3], [18].

As compared to HTML which addresses the presentation of data, XML separates presentation from data. Hence, XML specifies structures, not meaning. It is data-centric, not design-centric as it is concerned only with the document description and structure of data.

XML enables the easy interchange of structured data. It is based on the concept of documents having a series of *entities* (object). Each entity contains one or more *elements* (component parts). Information about the elements can be passed to other machines, enabling the sharing of data. Each of these elements can have certain *attributes* (properties) [13]. These attributes describes the way in which it is to be processed.

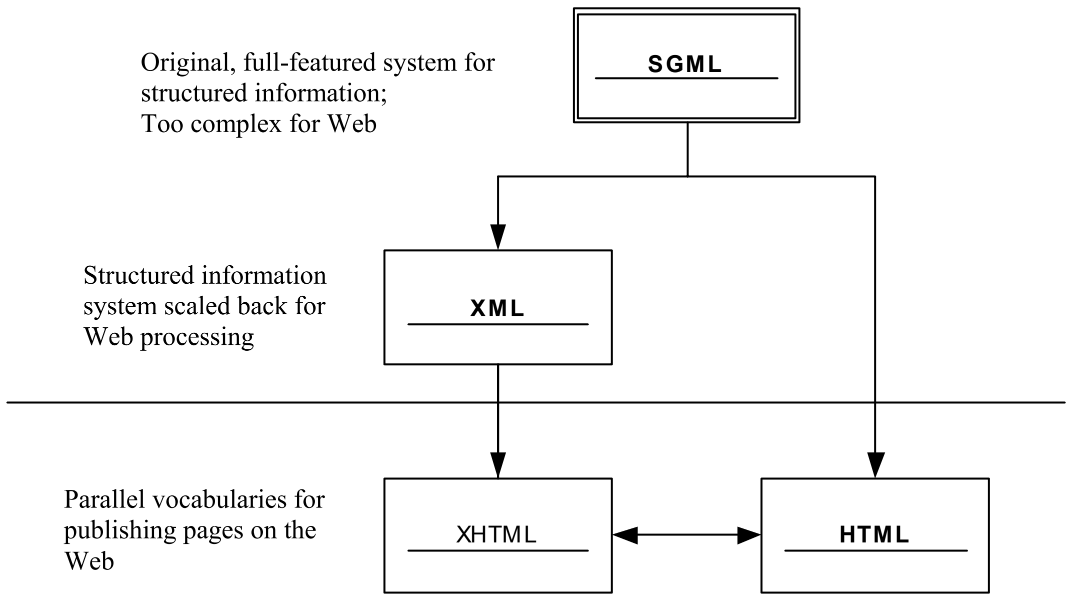
XML allows the user to create their own custom tags to describe their data. The tag would explicitly identify the kind of information contained in a document [25]. For example, the <ORDER> tag could identify a business transaction; the <PRICE> tags could contain the product cost, and so on.

The power of XML is on the backend, where data is passed to one or more systems. Communications among data sources and recipients is possible because XML acts like a "dynamic data dictionary," passing along data that has or can reference elements and attributes. But in order for this to happen, XML needs to be fed XML-based inputs [3].

To read an XML document, an XML parser/processor is required. This can be implemented as a browser (for display), or as an application module (for complex processing)

2.0 The XML History

The XML specification evolved from the Standard Generalized Markup Language (SGML). In 1996, the W3C started work to define an SGML-like standard, eventually named XML. Thus, XML is a simplified subset of SGML [16].


Original, full-featured system for structured information; Too complex for Web
SGML
Structured information system scaled back for Web processing
XML
Parallel vocabularies for publishing pages on the Web
XHTML
HTML

XML's core syntax, the XML 1.0 specification, became a W3C recommendation in 1998.

3.0 XML's Goal

The XML specification sets out the following goals [16], [28]:

1. XML shall be straightforwardly useable over the Internet. Users should be able to view XML documents as easily, and as quickly as they can view HTML documents.

2. XML shall support a wide variety of applications. XML should not be narrowly defined, such as only limited to a specific application. It should be extensible enough to create different applications; to extend to many applications, like content analysis, browsing, and document management

3. XML shall be compatible with SGML. XML should be compatible with existing technologies while offering new capabilities in delivering structured documents

4. It shall be easy to write programs that process XML documents. An assurance of XML's simplicity

5. The number of optional features in XML is to be kept to the minimum. This is to address incompatibility problems

6. XML documents should be human-legible and reasonably clear. A user should be able to read the document and figure out what the content means

7. The XML design should be prepared quickly. This is to meet an immediate need.

8. The XML design shall be formal and concise. XML must be amenable to modern compiler tools and techniques

9. XML documents shall be easy to create. Users need not have to use specific software to create documents

10. Terseness in XML markup is of minimal importance. XML markup need not be concise as concise tag names can be difficult to understand by just looking at them

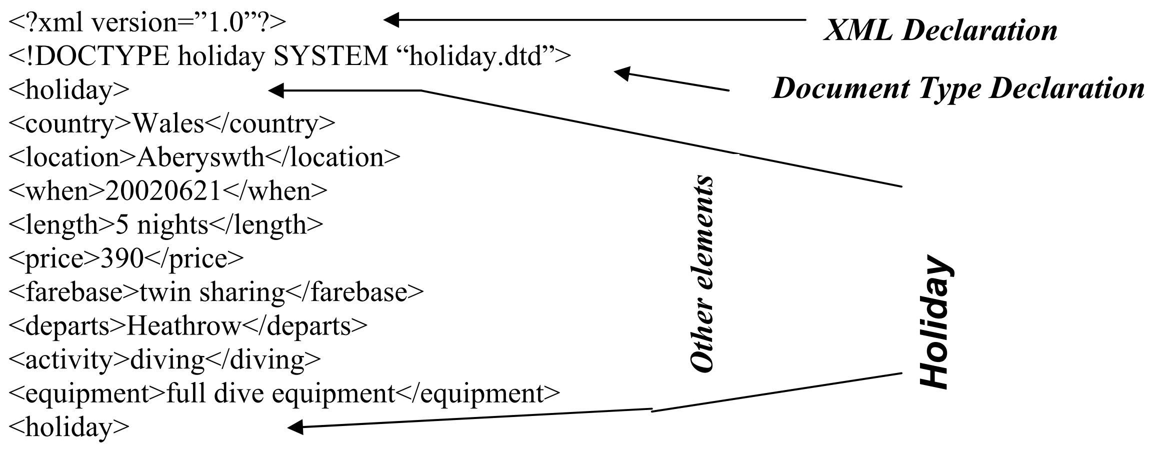
4.0 XML Document Structure

A good document structure enables users to make changes to the data easily. This is especially useful for automated data processing. To enable this, a XML document structure follows strict but simple rules.

The following are some requirements for any XML document [16]:

- Tags occur in pairs. The start and closing tags must be balanced.
- Closing tag has a forward slash preceding it. E.g. <student>David</student>
- Tags can be in upper or lowercase, but must have the same matching case for a tag pair
- Syntax rules must be adhered to. E.g. elements must be nested correctly
- Attribute values must be in open quoted
- XML documents can contain a Document Type Declaration (a !DOCTYPE statement) or this can be omitted. If included, it points to a file that contains the definitions of the structures expected in the document
- A document may only have one root element

An example of a document having only one root element, in this case "holiday", is as follows:



```
<?xml version="1.0"?>
<!DOCTYPE holiday SYSTEM "holiday.dtd">
<holiday>
<country>Wales</country>
<location>Aberyswth</location>
<when>20020621</when>
<length>5 nights</length>
<price>390</price>
<farebase>twin sharing</farebase>
<departs>Heathrow</departs>
<activity>diving</diving>
<equipment>full dive equipment</equipment>
<holiday>
```

An XML schema defines the elements that can appear within a document and the attributes that can be associated within an element. It defines the structure of the document, including the relationships between the elements, the data type the elements can store, and how many of that same element can occur [19].

5.0 Schemas

A XML document structure is described in a XML schema. XML documents that adhere to the vocabulary defined in a schema are considered ‘valid’.

There are two approaches to defining XML schemas. They are:

5.1 Document Type Definition (DTD)

A DTD is a set of rules that defines the structure for an XML document. Using special syntax to describe the structure of XML vocabularies, DTDs serve as templates to pour data in and arrange it.

DTDs can be shared across networks. This enables users to read each other’s files, enabling data exchange. DTDs can reside in the document, or be kept in a separate file referred to by the file that conforms to the DTD. Linking to a separate DTD is usually used when there is a large set of XML documents that must conform to the same DTD.

XML does not require the presence of a DTD. If no DTD is available (e.g. because the user did not create it), the system can assign a default definition for undeclared components of the markup [16], [19].

5.2 XML Schema

Like DTDs, the XML Schema is used to specify the schema of a particular class of documents. Using XML syntax, users need not learn a new language to define the grammar of XML documents [16]. The user just needs to declare attributes and elements using the XML Schema.

6.0 Document Validity and Well-Formness

XML documents follow two rules of syntax [5], [16]. They are:

1. A document is considered *valid* if it conforms to the rules in the DTD. That is, if it follows the DTD rules, it conforms to the XML specification. This also means that it is *well-formed*. A well-formed document adheres to the XML syntax rules

2. If the document does not conform to a DTD, but conforms to the XML specification, then it is said to be *well-formed*

Document validity is useful because as valid documents, that are much easier to use. *Invalid* documents can contain any set of tags, and in any random order. In such cases, document reusability would be problematic [19].

Another characteristic of valid XML documents is that they are also compatible with SGML. Hence, cross-platform interoperability is assured.

A well-formed document is easy for computers to read. A document is “well-formed” if [16]:

- Its syntax conforms to the XML specifications
- The start-tags and closing-tags match up
- Elements are properly nested
- Empty tags use the special XML syntax (e.g. <empty/>)

- All the attribute values are nicely quoted
- All the entities are declared
- There are no external entity references

7.0 XML's Core Elements and Functionality

The XML specification addresses the "core" language itself, which describes [3]:

- the syntax
- the styling requirements
- the linking capabilities (to logically connect documents and data points together)

XML is hierarchical in nature in that it defines the structure of the information, such as family tree with parent/child relationships. It permits users to define or create their own tags and elements and grants nesting [18].

XML defines its grammar for others to use. It provides a formal syntax for describing the relationships between entities, elements and attributes that makes up a XML document. The syntax is used to tell the system how it can recognise the component parts of each document. XML is strict about its syntax [29]. E.g. all the markup tags used must consists of matching start and end tags, like: **<name>**Fred**</name>**

By defining the role of each element of text, users can check if each component of the document occurs in a valid place. E.g. it checks that users do not accidentally enter a third–level heading without having first entered a second-level heading.

Although XML is able to do many things, XML is not [13]:

- a predefined set of tags. XML does not define these tags itself. It essentially lays down the procedures for defining them
- a standardised template for producing particular types of documents
- a standardised way of coding text

8.0 Extensibility

XML's strength is its ability to describe and structure data. This means that XML can be used to define (to extend) to other vocabularies. That is, XML-based languages can be "extended" to support unique and diverse data structure requirements, such as commerce, chemistry, and cooking recipe [5]. This extensibility of XML comes from its metalanguage capabilities.

Hence, any XML user can define the data descriptions and structures to his requirements. And this data description can be shared with others.

9.0 Metalanguage and Metadata

A metalanguage is a language that is used to describe other languages. XML is a metalanguage as it is used to define a set of data element tags, vocabularies and applications.

Metadata can be defined as "information about the data". In XML's context, the metadata describes data element tags or attributes. The XML metadata can tag data to provide meaning, content, or context [1], [5].

In a XML document, DTDs are used to define what these tags mean, and what the data elements represents. For example, the number "0":

- could mean a temperature reading, using a temperature tag, *<Celsius>0</Celsius>*;
- or, the balance in a bank account, using a dollar tag, *<Dollars>0</Dollars>*

Although both values are zero, using the respective tags, one can now tell the difference and perhaps manipulate them accordingly.

10.0 The Importance of Metadata

Data and information are valuable and critical assets to most organizations. Unfortunately, in most legacy systems, these assets are stored in data repositories that does not support easy data access or retrieval, nor allow easy cross-platform data sharing or exchange [25].

Using XML metadata capabilities, data access or exchange is possible for legacy systems. Such systems can exploit XML tags for database access, or for migrating into a new capability without having the need to use additional software [5]. This means that without the need for modifications, any application can use the same data.

Hence, metadata presents a competitive advantage to users as it facilitates the use of the already-available data. With metadata, seamless data sharing and transfer can be done.

11.0 Outputting XML Documents

XML documents are output-medium neutral as no output control is defined in the documents. Users can use additional technologies to associate output controls and formatting to the XML tags and structure. Output possibilities include browsers, language processors, mobile devices, printed page, etc. To illustrate how powerful and diverse this capability can be, consider the scenario of picking up XML-enabled emails as voice messages through the mobile phone [29].

A wide range of output support exists to handle output formatting. For example:

- Cascading Style Sheets (CSS)
- Extensible Style Sheet Language (XSL)
- Extensible Style Sheet language for Transformations (XSLT)

12.0. Transforming and Re-Formatting Documents

XML supports data transformation and re-formatting. Transforming data from internal formats to those acceptable for data exchange is important. For example, two parties may want to electronically exchange data [19]. Using XML, both parties can store the data that meets their own specific requirements, and yet extract the common elements for data exchange.

XML transforms data easily because XML separates document content and structure from output formatting. It is the underlying data that is subjected to the transformation processes [16]. Furthermore, data transformation can extract a subset of information from a document by using technologies that can navigate document structures and locate specific elements that meet certain criteria.

For XML data re-purposing, customizable data elements can be defined for the purpose of changing the look and feel of, say, a Web page. Or, a Web page can be re-purposed to work with other devices such as wireless devices or text-to-speech systems.

The ability to easily transform and re-purpose data results in lower costs because [31]:

- Data does not need to be recreated each time it is used
- By using XML to deliver customized Web pages, the user does not have to incur additional costs of detailed scripting or programming

13.0 Contrast between XML and HTML

XML data is text-based information that has context. The enables XML data to be read by any application and the context adds meaning and usability to the text. In contrast, HTML data is primarily free of text in that HTML text is used only for display purposes, not for processing.

XML users can define their own tags and element, nesting and grammars, and they can share those tags with other users. Users can create DTDs or Schemas that defines the document structure, as well as logical elements, tags, and attributes. The DTD provides instructions for parsing the document accurately. After the document is parsed, an XML application can use the data for display, data processing, manipulation, transformation, and other capabilities that depend on the data. This is the core concept behind metadata. With no modifications, any application can use the XML data [5].

XML users can embed data in a document without limiting names or specifying the order. The data element tags, as defined by the DTD, serve a similar purpose as, say, the tables in a database. When a user uses a XML document, he can decide on a level of detail for its DTD and document structure. And other users can build on top of the standard, or ignore details as appropriate [13]. Hence, if Toyota creates a DTD for a document that only uses <customer_name>, Ford Motors can use the same document and add tags for <first_name> and <middle_initial>. Toyota can use the new document and include this new capability in its applications. Or it can continue to use its existing applications, which will ignore the new tags with no significant loss in processing.

In summary, XML differs from HTML in these main areas [5], [33]:

1. HTML defines how elements are displayed. XML defines what those elements contain.

2. HTML uses pre-defined tags. XML is extensible, allowing tags to be defined by the user. New tags and attribute names for data can be defined at will. Data can be structured not only in accordance to formal criteria (such as document header, body text, etc.), but also by referring to its contents.

3. Searching HTML documents produces inaccurate results because no context exists for the data contained in it. XML allows data to be searched more accurately and easily as XML provides the context.

4. Unlike HTML, XML is self-describing. This means that an application can interpret XML data without prior knowledge of its data structure.

One limitation of XML is that binary information cannot be embedded in the body of the XML document. On the other hand, encoding information in plain text with non-proprietary tags might be better than using proprietary and platform dependent binary formats.

14.0 The XML History

Please refer to Appendix A

15.0 The XML Standards

Please refer to Appendix B

16.0. The XML Landscape

Please refer to Appendix C

Chapter 4

XML Features, Benefits and Capabilities

Information and data interchange is the cornerstone for many business applications today. This chapter describes the features, benefits and capabilities of XML from a management's perspective. This is by no means a complete list as new features and capabilities are being explored and discovered. But it covers the most basic and important ones.

1.0 Introduction

Most business transactions require the need to exchange data, information and documents. Often, data and information will have to be prepared, transmitted, tracked, queried, etc.

XML makes possible many tasks that are previously not possible with HTML. XML is designed to be powerful, yet easy for users to manipulate. For example, data exchange could be a nightmare if the data has to be massaged, manipulated, and translated for each new application. Consider the situation if data needs to be transferred from a catalogue → to an invoice → to an inventory → to a bill-of-materials. It would require at least three different steps to re-format the data for each specific application. With XML, data for all of the above applications can be easily reused without re-formatting the data each time. There is no necessity to pre-agree on rows, columns, syntax, or other data formats, because all that information is included with the XML document [18], [3].

2.0 The Features, Benefits and Advantages of using XML

Some of the features, benefits and advantages of using XML are listed below:

2.1 Universal Data Exchange Format

XML is a standard for defining data across application or industry domains. Data can be shared across many sources.

Although, many computer standards are cross-platform and cross-application, data from one platform or application may not work on another platform or application without the need to do some intervening data massaging [5]. XML eliminates that extra step by providing a universal data transfer format that is platform and application independent. This enables interoperability and easy application integration. It also means that the application is able to work with legacy and future systems. For example, a XML-enabled database can be integrated with existing database applications. The XML document that is not in database format could easily be represented in a database format [29]. Vice-versa, if the components of an XML document are stored in a database, the results of a database query could be presented as an XML document.

2.2 Self-Describing Data

A universal data exchange format alone does not guarantee interoperable access of data. Data exchange applications require uniform standards for communications [5]. Such standards include the data element types and a common definition of the element types. The use of metadata in XML addresses this requirement. For example, in traditional databases, data records typically require schemas to be set up. However, by using XML metadata in the form of tags and attributes, documents can be stored without such schemas. As the tags are self-describing, when reading a XML document, a user would know that the number "60" in **<AGE>**60**</AGE>** refers to "age" and not, "weight". The meaning of a number is clearly and unmistakably associated with the number itself.

Many legacy data are lost simply because there is no documentation on how one actually reads the data. For example, a Lotus 1-2-3 file on a 15-year old, 5.25-inch floppy disk may be irretrievable, not without investing time and resources to retrieve the data. And data in even lesser-known digital format may be lost forever. XML removes this possibility [16]. Suppose the following XML code fragment has been found. It has survived the ravages of time.

```
<PERSON ID="p1100 " SEX=="M ">
<NAME>
<GIVEN>Andrew</GIVEN>
<SURNAME>Ang</SURNAME>
</NAME>
<BIRTH>
<DATE>27 April 1918</DATE>
</BIRTH>
<DEATH>
<DATE>24 November 1999</DATE></DEATH>
</PERSON>
```

We can have a good idea that this fragment of codes describes a man, whose name is Andrew Ang, whose date of birth is April 27, 1918, and who died on November 24, 1999.

Hence, self-describing data ensures usability through the years. Most commercial entities have the legal obligation to store their business records for a certain period of time. And if required, they must be able to reference the records very reliably and quickly. But how will they ensure their digitised records will remain understandable, say, 10 years from now? XML's self-describing format solves this problem as the content, relationship, and meaning of any data can be captured [5]. The information will remain understandable even if the applications that created it are long gone and forgotten.

2.3 Structured Information

XML's format supports structured information. When data is created using an XML editor, the user not only input the data, but also defines the structural relationships of the data. This means that document structures can be nested to virtually any level of complexity [5].

Being structured, XML facilitates the use of context and meaning. The tags, attributes and element structure provide context information to interpret the meaning of the content. Users can specify a vocabulary that defines the elements in the document, and the relationship between the elements. Users can define their own tags and the structural relationship of the data elements with the help of a DTD or schema [16]. For example, a XML search engine would be able to understand the term “Lotus”, and allow car lovers, or flower-lovers, to retrieve two entirely different sets of documents based on the context of the query.

XML is able to handle large, complex, tree structures. Furthermore, users can selectively update individual elements (granularity) or whole subsections, rather than having to republish complete documents. In contrast, a HTML Web page must be refreshed each time there is a change, no matter how minute [19].

2.4 Extensibility

XML is extensible in that users can create entirely new tags, modify existing tags, define new meanings, and share them with others. Specific document vocabularies can be created.

The extensibility of XML makes it relevant to industries, and application types. It means that users have full control when customizing their data, because there is no predefining of any tags. Users are able to exchange data without worrying about whether or not the other person can receive the data, or whether the person has the particular proprietary software that was used to create the data [25].

To read or modify documents, neither the sender nor the receiver will need special support like complicated plug-ins. A sender can send documents to people outside of the industry, knowing that the recipient will at least be able to view the documents. This extensibility makes XML a good fit when getting different systems to work with each other [19].

2.5 Open non-proprietary, vendor-neutral Standard

XML is an open, non-proprietary, vendor-neutral standard. Interoperability between old and new, as well as heterogeneous systems is ensured. Being an open standard, users from one company can create an application and data that other companies will be able to use.

As a non-proprietary standard, XML is not encumbered by intellectual property restrictions. Users are protected from vendors who often change their proprietary standards [16]. For potential companies making an investment decision in XML technology, this is reassuring as it maximizes the lifetime of their investment.

Users can use any tool that understands XML. There will be many commercial tools available to help users to author, manage, and deliver it. Users will not be locked into a particular application simply because that is what their data is already written in, or because it is all their receivers can accept [29].

2.6 Loosely Coupled Architecture

XML contains data and the metadata about the data. It does not specify any specific way in which the data should be processed. It does not put any limitations on how the data is to be handled or displayed. The data sender does not need to know the details of how the receiver intends to process the data. Hence, it has a loosely coupled framework.

Systems that are built on a loosely coupled XML integration framework are future-proof because changes to the computing environment will not affect XML data exchange. XML documents can be freely exchanged across different platforms and applications as long as the applications are XML-enabled. Any XML-enabled application can access and manipulate the data contained in the XML document [11], [19].

3.0 Capabilities

Following is a summary of some of XML's capabilities:

3.1 Multi-Dimensional Document Processing

A plain text file is considered one-dimensional. The user can **read** the text file. A HTML text file is two-dimensional. The user can **read** and **display** the file. A XML file is a multi-dimensional document. The user can **read**, **display**, and **process** the file in multiple applications.

If a user migrates from a flat text, through HTML to XML, the degree of utility provided by adding different types of markup to the text increases dramatically. Consider a relational database management filing system (DBMS). In its native form, most database technology is not able to handle data such as documents, image, audio, digital signatures, etc. [19]. Typically, DBMS tends to 'flatten' the world of data into tables (2-dimensional structures). But the real world is not flat, nor 2-dimensional. It is full of complex relationships between different types of files, references, hierarchies, etc. Hence, it would not be possible to store the data in the DBMS.

XML offers a solution as it able to handle multi-dimensional data. Users can store, retrieve, manage, search, and distribute them from a single source, while at the same time, preserving the relationship of the data [13].

3.2 Textual and Language Independent

XML is able to support Unicode and multilingual documents. It has built-in support for texts for all the alphabets. Being language independent, it does not need standard binary coding or storage format [5].

Language independence fosters immense interoperability between heterogeneous systems.

3.3 Data Validity Checking

As XML documents come with built in error and validity checking, it provides the means to validate data formats to ensure content correctness.

XML users can specify the vocabularies to define the elements in a document, and the relations between these data elements. As users define their own tags and create the structural relationship of the elements, XML parsers can be used to check the validity and integrity of the data. This makes it easy to validate the structure and content of the document. Provided that a schema has been referenced and adhered to, the XML parser is able to provide content validation by ensuring that all the required fields are provided and in the right order [3].

For example, a XML document may contain a description of its vocabulary for use by applications that need to perform validation of the data contained in that document, before, say, populating a database. A XML Schema is used to verify that all elements are correctly specified, in the right order, and that values fall within acceptable pre-determined ranges.

With XML parsers, application codes can be quickly developed. It allows users to easily generate XML document as XML parsers allows users to code faster by giving them a parser for all their XML documents [11].

3.4 Vocabulary Conversion Capability

XML enables automatic, repeatable transformations of XML vocabularies. Transformation from one XML vocabulary to another is a commonly required for data exchange and application integration. Some examples of transformation are [3]:

- To convert name tags
- To selectively extract data
- To convert data types

Some transformations can be more complex. For example, if one system holds ‘Date of Birth’ and another requires ‘Age’, some computation must be applied to produce one from the other.

3.5 Easy creation of different views of the same document

As XML tags describe the meaning of the data and not the presentation, the look and feel of a XML document can be easily controlled (by using XSL style sheets). This means that the look and feel of a document (e.g. a web page) can be changed without touching the content of the document [12]. Different software is not required in order to get a different view of the data.

Multiple presentations of the same content can be easily and quickly rendered. Users can get different ways of looking at the document as each time a different style sheet is used to the same document, a different view is obtained. Furthermore, using style sheets, users can define mechanisms for sending data to any type of output device. They also have granular control over what information is to be published [5]. With XML style sheets, the cost and time it takes for users to deliver information to whomever they need, and in whatever appropriate output format will be reduced.

3.6 Efficient Server-Side Processing

XML is able to produce documents that functions independently of the server. For example, users can download and manipulate a document off-line without involving the server.

A XML document is like a container of structured information. Once downloaded, it can be re-processed locally. Users can process the data at any point. This is unlike a HTML document, which presents a fixed view to the user. It must be re-generated at the server-side if this view is to be changed [29].

3.7 Flexibility of Content, Code and Formatting

XML documents can be visualized in terms of associations between content, code, and output formatting. Each of them takes a number of forms [31]:

- Users can use the whole content, or selectively extract some of it, or selectively update or transform other parts of the content
- Users can use different programs with the content
- Users can at anytime apply appropriate output formatting, based on the output medium in use

3.8 Data Comparison and Aggregation Capability

The XML document tree structure enables documents to be compared and aggregated efficiently, element by element.

4.0 Conclusion

Although XML offers many benefits to the user, there seems to be a lot of hype surrounding the use of XML and its perceived benefits. It is said that XML technology is not revolutionary, at least in the approach to solving general computing problems [31]. Problems such as interoperability of code and data, both across and within, different platforms and application boundaries have been around for many years.

Therefore, it can be said that XML technologies, by using the most successful strategies and techniques that have been improved and refined over the years, has provided the solution to these general computing problems. Hence, this has made XML better positioned to handle new problems, and making it an obvious choice for data exchange.

Chapter 5

XML Security

The adoption of XML systems requires the use of XML security mechanisms to address security requirements such as data privacy, authentication and confidentiality. This chapter describes:

- *The need for security for XML documents*
- *XML and security*
- *Uses of XML Digital Signatures*

1.0 Introduction

One basic goal of XML is to enable the sending, receiving and processing of data. As with other technologies, such data transfer require effective security mechanisms for XML-enabled documents. However, existing security mechanisms like Secure Socket Layer (SSL) do not sufficiently address some of the security requirements for XML.

Some of the general security requirements of are [30]:

1. Ensuring long-term authenticity
   E.g. Who sent them? Did the document actually come from the purported sender?

2. Protecting data confidentiality
   E.g. No one else can access or copy the data

3. Ensuring data integrity
   E.g. Has the data been modified in transit?

4. Supporting non-repudiation
   E.g. Can the sender deny sending them? Can they deny the contents of the data?

In a typical XML transaction like the issuance of e-receipts, the transaction requires some degree of authentication. And if transactions involve multiple parties, different parts of a message may need different types of authentication, for different recipients [22]. For example, in a purchase order, the payment portion from a customer could be extracted and sent to the Finance department, and then to the bank. Likewise, instructions and messages relating to the purchase may need authentication at the Purchasing department as a protection against forgery.

Ensuring data confidentiality is another important requirement. As with authentication, granularity below the document level is often required [23]. For example, staff residence phone numbers could be less confidential than say, salary information. Or, a customer's credit card number is more sensitive than his age. Hence, by encrypting different fields with different keys, the fields can be secured to different classes of recipients.

2.0 The need for Security for XML Documents

Non-XML security mechanisms do not fully ensure secured transactions for XML documents. While generally adequate for low-value transactions, most of these security mechanisms do not provide the enhanced security or flexibility as would be required in high-value transactions, or protection of the sensitive data that was transmitted [23]. While security requirements such as confidentiality and authentication can be obtained by using mechanisms such as Secure Multipurpose Internet Mail Extensions (S/MIME) or Pretty Good Privacy (PGP), using them will require additional non-XML mechanisms that may have different XML concepts. Furthermore, such mechanisms may clash with the XML system.

For point-to-point security between the sender and receiver, one can use non-XML mechanisms like the Secure Sockets Layer (SSL), IP Security (IPSEC), or Transport Layer Security (TLS). SSL for example, provides for the secure interchange of data between a browser and server. But once received, the data is usually left unprotected on the server. But, the important point to note is that SSL protects the data in transit, when its confidentiality is comparatively, far less likely to be attacked [30]. Sniffing IP packets in transit to obtain a person's credit card number is not as efficient as breaking into a database containing thousands of credit card numbers.

This problem is multiplied where a message is routed from server to server. If the data itself were encrypted, instead of just its transport, it would help reduce the incidents of unencrypted data left vulnerable on the servers.

Furthermore, non-XML secured channels provide only one level of confidentiality for all material sent through that channel. They have limited authentication provisions. After data have been stored, the data [2]:

1. has no confidentiality because it was decrypted as it exited the secure channel
2. has no integrity protection
3. are not associated with any authentication from that secure channel that could be forwarded to or recognised by a third party

Hence, effective XML security mechanisms are required to support the authentication and confidentiality of documents, or sections of documents.

3.0 XML and Security: Digital Signatures

One advantage XML brings to security is the ability to integrate with other XML data vocabularies. For example, an authentication protocol in XML may contain information from another XML vocabulary [26]. The advantage of different security protocols using XML, instead of separate binary formats, is that the protocols will be able to better understand and work with each other.

However, the very features making XML powerful for commercial transactions (e.g. features such as being semantically rich, structured data, text-based, etc.) provide both opportunities and challenges for the application of digital signature operations to XML-encoded data. For example, consider a situation where a XML document flow between personnel of a company, and where a digital signature implies some sort of commitment or assertion. Here, each personnel may want to sign only that portion for which they are responsible for and assume some level of liability. Non-XML-enabled signature standard does not provide syntax for capturing this type of high-granularity signature, or the mechanisms for expressing which portion a personnel wants to sign. However, XML signature standard allows such granular flexibility [2].

Key to developing effective security mechanisms is the placement of security within the architecture. Within a network protocol stack, security mechanisms can be placed at many different levels. Generally, security placed at a lower layer results in a better end-to-end security solution [22]. This is because all the information in the layers above is protected. However, the higher the level, the easier it is to combine the security data with the application. Furthermore, a finer granulation of security can be achieved.

A portable security mechanism is one where data can be stored and reused in different protocols As XML is always the top layer of a protocol stack, XML security mechanisms are more portable than lower level security mechanisms. As long as domains understand and are able to process the necessary XML vocabularies, they can make use of the same XML security mechanisms. E.g. a XML signature is retained by default, as they are part of the application data structure. But security data structures that are built into the sockets' API or the IP headers are generally discarded before being processed [15].

Generally, security at the IP layer (IPsec) can prevent attacks such as denial of service and create virtual private networks. Security at the socket layer (SSL) can ensure data confidentiality. It provides authentication for server and client using the Diffie-Hellman protocol. Hence, SSL authentication and encryption can provide integrity for non-portable data [26].

4.0 Uses of XML Digital Signature

XML digital signatures could be used in the following applications [30]:

4.1 Digital signature of Web page

By signing a Web page, users will be ensured the data conveyed is authentic. For example, when a document displayed as a web page is signed, it indicates that the document is genuine. This method is preferable to obtaining the document using a secure link channel to an authenticated server, because it is the document that is to be authenticated, not the server, nor the secured connection. Additionally, the document can be placed at a different location and still retain the security of the digital signature.

4.2 Trust mechanisms that are machine interpretable

If the XML digital signature standard is combined with a logic-processing language, the decision can be done by the machine based on the logical rules set up by a client where the trust decisions have been laid out logically by the client machine [26].

For example, a decision to download code from an unknown company might pose a security threat. As the signed code and the public key certificate connect code to organisation, the client machine could gather signed statements on the company from various sources like auditors and business analysts, to determine the company's financial health. The accumulated information could then be presented to the client for decision-making.

4.3 Security protocol using XML Signature

If there is an authentication protocol that results in the authenticated client receiving an authenticated token at the end of the process, the token can be used to verify these credentials to other parties. An XML-enabled token can contain data created using the multitude of XML vocabularies [24]. The XML signature of the data residing in the token will indicate the authentication server's security approval.

5.0 Conclusion

XML security mechanisms must ensure data authentication, access control, confidentiality, integrity and non-repudiation.

- Authentication and access control separate users who are allowed to perform certain actions from those who are not allowed
- Confidentiality ensures that messages are kept secret between sender and recipient
- Integrity ensures that messages cannot be altered without being noticed
- Non-repudiation services prevents parties from the denial of an action

These are all required by users to prevent fraud, abuse, or actions not permitted by the user.

Chapter 6

XML Digital Signature

The adoption of XML for doing business across the Internet requires the same, if not better, security guarantees as the real world. This section describes the use of XML digital signatures. The focus is not on the nooks and crannies of the specifications, but on the basic reason for its existence, the fundamental properties that define an XML signature and examples of such signature.

1.0 Introduction

XML signature standards are digital signatures designed for use in XML transactions. The standard defines a schema for capturing the result of a digital signature operation applied to arbitrary digital data. It is a specific messaging syntax. The signature object itself appears in XML syntax and makes use of a number of XML standards to define its precise XML format.

The objectives of XML digital signature are:

- to ensure in-transit data are complete and accurate (for data integrity)
- to provide a mechanism to control and manage the data that is passed and presented

Alternative mechanisms such as SSL, protects data in transit. However, in the case of SSL for example, it does not meet the requirements of XML data that are not in transit. Hence, the reasons for using XML signatures are to meet some of these requirements [23]:

- Portability.
  XML signatures are portable. Entwined within the XML data, the signature forms a constituent part of the data. It is not a sub-layer that will be stripped off as the data is retrieved from the network
- Flexibility and granularity.
  XML signatures can be made to refer to parts of a document, or to many documents.

A XML digital signature is a rich, complex, cryptographic object. The signature syntax is designed to provide high degrees of flexibility and extensibility, and is conducive to almost any signature operation. However, as it relies on a large number of disparate XML and cryptographic technologies, the culmination of such technologies results in a signature syntax that can be quite abstract and daunting [2].

A non-XML signature results in raw binary output of a relatively fixed size. For example:

- RSA signature, the output is related to the key-size used
- DSA signature, the output is related to the representation of the encoding used

But instead of raw binary digital signatures, XML signatures are related to messaging syntax (e.g. PKCS). The structure is specified in relation to the source documents. For signature verification, it can encompass a cryptographic key or X.509 certificate.

A XML signature is a XML document. It has all the properties of a well-formed XML document. All the information needed to process the digital signature (e.g. the verification information) is embedded within the signature representation. Except for the actual signature and digest values, all the elements and attributes are processed as “normal” XML [15].

XML signatures require minimal processing, even if the applications do not have XML signing or verification capabilities. There is no complexity in the signature process or cryptographic operations used. The added complexity lies in the additional processing features demanded by XML-enabled documents.

To verify a raw digital signature, the signer must provide to the verifier, information such as the type of algorithm used, recipient details, and the verification key. Once these parameters are configured, it is usually difficult to change them.

Even before XML, solutions are available for extensible and flexible processing, For example, ASN.1 and BER encoding, together with a hierarchical set of object identifiers. The algorithm object identifier is a unique bit-string, encoded in raw binary format that conforms to BER/DER. It is used to identify a type of signature algorithm [15]. However, this method of compact binary representation to inform the other party the type of signature algorithm to be used during the application processing can be a tedious and complex process.

So rather than using the algorithm object identifiers, being a text-based format, XML Signatures uses a text-based algorithm identifier to denote the same RSA signature with the SHA-1 hash function (http://www.w3.org/2000/09/xmldsig#rsa-sha1).

Although the non-XML algorithms object identifier is more compact and smaller representation, the pervasiveness of XML parsers makes processing text-based identifier more viable and much simpler.

XML Signature has tried to avoid compact binary representations, although for reason of backward compatibility, the binary-coded identifiers are sometimes still used in the creation of the signature [14].

2.0 XML Signature Semantics

A XML Signature associates the contents of a signature manifest with a key via a strong one-way transformation. That is, the signature defines a one-way signature operation based on a signing key.

Like normal paper signatures, digital signatures have wide applications for associating data (or document) with an individual. In the case of XML Signatures, while this trust semantics is useful and practical, it does not by itself associate a signing key with an individual. Instead, the XML Signature provides the means to establish the association. Establishing the association is done by packaging the verification key within the XML Signature, via an element. The XML Signature presents the verification material (either the public key, or the certificate containing the public key) to the application, leaving the issue of trust to the application [23].

Mechanisms to validate the identity of a signer based on public key information already exist, e.g. the certificate path validation. However, the decoupling of entity verification from the actual signature provides the application more flexibility in deciding its own custom trust mechanisms, for example, to check if an entity has the authority to sign a document, or a portion of a document.

It must be noted that not all private keys are authorised as signing keys. Also, a trusted authority might have restrictions on private key usage for a particular individual, or, an individual's key might have been revoked altogether. These additional trust semantics lie outside the scope of the XML digital signature [14].

Just like other documents, XML documents can be digitally signed in its entirety (as a whole). However, difficulty arises if only parts of a document need to be signed, and/or by different users, or when this needs to be done with selective encryption. It may not be practical to mandate a particular sequence of sectional encryption by specified people acting in order, as successful processing of the different parts of the document will depend on knowing this.

Furthermore, as a digital signature asserts that a certain private key has been used to authenticate something, it is prudent that a signer view the item to be signed in plain text. This may mean having to decrypt part of something already encrypted for other reasons. In some cases, encrypted data may have to be encrypted further, as part of a larger set [23].

The more the different possibilities are considered in sets of transactions involving a single XML document, the more the potential for complexity (e.g. a series of records used in a workflow sequence, processed by many different applications and different users)

3.0 XML Digital Signature Standard

XML signature standard defines a schema that captures the result of a digital signature operation when applied to XML data. This standard defines the syntax required to sign either all, or a part of a XML instance.

XML digital signatures have the flexibility to sign only specific portions of the XML tree, as opposed to the complete document. This flexibility is useful when [15]:

- the XML document have different components that are authored, at different times, by different parties, with each party signing only those elements relevant to them

- ensuring the integrity of certain portions of the XML document, while leaving open the possibility for other portions of the document to be changed later. For example, a signed XML document is delivered to a user for editing. If the signature were over the full XML document, any changes made to the default document values would invalidate the original signature

However, this flexibility, does not lend itself well where a misplaced space results in a completely different fingerprint that is unverifiable. To address this, the use of canonicalization (XML-C14N) has been defined in a W3C document called Canonical XML. It follows a pre-identified set of processing rules to structure an instance into its simplest form [14]. The rules are to ensure that instances are structured the same way, every time. Hence, the signature will not be confused due to stylistic differences, such as misplaced spaces.

4.0 The Components of an XML Signature Element

The components of a skeletal structure of a Signature element are as follows [24]:

```
<Signature>
  <SignedInfo>
    (CanonicalizationMethod)
    (SignatureMethod)
    (<Reference (URI=)>
        (Transforms)
        (DigestMethod)
        (DigestValue)
     </Reference>
 </SignedInfo>
 (SignatureValue)
 (KeyInfo)
 (Object)
</Signature>
```

The explanations to some of the main components are as follows [24]:

Component	Explanation
(<Reference (URI=)>	Each resource to be signed has its own <Reference> element, identified by the URI attribute
(Transforms)	The <Transform> element specifies an ordered list of processing steps that were applied to the referenced resource's content before it was digested
(DigestValue)	The <DigestValue> element carries the value of the digest of the referenced resource
(SignatureValue)	The <SignatureValue> element carries the value of the encrypted digest of the <SignedInfo> element
(KeyInfo)	The <KeyInfo> element indicates the key to be used to validate the signature. Possible forms for identification include certificates, key names, and information

5.0 The Signature Generation Procedure

XML Signature uses canonicalization processes to format a XML document into a standard format. The signature is generated from a hash over the canonical form of a signature manifest. This "manifest" is expressed using XML, and is a list of resources to be signed.

Syntactic variations over logically equivalent documents are allowed in XML. Hence, it is possible for two XML documents to differ, but still be semantically equivalent. For example, the addition of one white space inside XML start and end tags [4].

E.g. 1: **<Reference URI= "ManifestList"/>**

E.g. 2: the element modified by adding a space: **<Reference URI= "ManifestList"/>**

For hash functions, this will be a problem as they are sensitive to single byte differences. In both examples, if a SHA-1 hash is applied to the elements, the SHA-1 hash output will produce different octet-strings. Although the hash values do not match each example, the semantics of the empty XML element (e.g. 2) are exactly the same (e.g. 1).

Any change to the message to which a cryptographic hash algorithm is applied will result in a different value. This ensures confidence as to the integrity of the message. This is acceptable for normal use. However, complications will result in situations where two XML documents differ in exact textual comparison, though they may be logically equivalent.

To remove the syntactic differences from semantically equivalent XML documents, a canonicalization algorithm is applied to the signature manifest. This algorithm is used to ensure that the same bytes are hashed and signed subsequently [4]. Hence, this solution provides for a more robust normalization algorithm within XML Signature processing.

6.0 Hash Functions

When applied with digital signatures, hash functions are convenient to use. It reduces the size of what is being signed and it speeds up the signing process.

When XML Signatures are generated, hash functions are used in two scenarios: In the manifest, each resource is hashed. The collection of resources is hashed a second time during the signing operation

The manifest is especially useful if the number of resources to be included increases. Without a manifest, applying a signature algorithm to each resource is time-consuming as it hinders the creation and verification of an XML Signature. So, instead of signing each resource, each resource is hashed. Then, the hash value and resource location is included in the manifest, resulting in a much faster process [24].

7.0 Canonical XML Specification

The canonicalization process is used to create a XML document into a standard format. The canonical specification describes a method for generating a physical representation of a document (known as the canonical form). Permissible variations is allowed, ensuring that if two documents have the same canonical form, these two documents are to be considered logically equivalent within a given application context.

The XML canonicalization process is to ensure that instances are structured the same way, every time. Documents having the same intrinsic information will have the same binary representation, and hence, the same signature. The signatures will not be confused due to the stylistic differences (e.g. extra or misplaced spaces). It allows the precise binary form of the original digital signature to be recaptured when validating, without requiring the retention of the original XML document [15].

This is useful in digital signing as textual variants that have no logical change implications, should not automatically mean that the integrity of a document, or that the authentication of the sender is suspect. This is something that might occur as a result of different treatment by different tools (e.g. parsers) generating different texts and, in turn, different message digests. During both signature generation and validation computing, the message digest should be done on the canonical form. If the digest matches, it confirms that the canonical forms over which they were computed also match, even though the textual forms may differ [14].

8.0 The Information in XML Digital Signatures

The XML digital signature can be divided into four parts [2]:

(1) The description of what is being signed
(2) The digital signature itself
(3) Key information (optional), and
(4) Other relevant information (optional)

The operation must provide information to ensure that the signatures are verified.

The information in a signed XML instance contains [14]:

1. The Canonicalization Method
   This identifies the rules and structures an XML instance prior to signature. It ensures the proper form of the data contents being signed, so that the verification algorithm will succeed if the data contents have not been modified

2. The Signature Method
   This identifies the algorithm used to sign the message digest.

3. The Digest Method
   This identifies the algorithm that creates the message digest signed by the signature method. It ensures that data contents are processed with the same algorithm when comparing the resulting values.

4. The Digest Value
   Contains the message digest (the fingerprint to the data contents being signed), i.e. the fixed-string output when data is processed through a message digest algorithm.

5. The Reference Information
   This refers to information about the data contents.

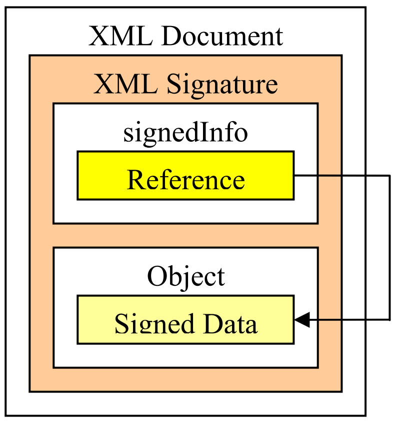
6. The Signature Properties (optional)
   The purpose is to add context to a signature. E.g.: serial number. The items can be included in the reference information.

The core generating process of signing a XML instance begins with the canonicalization method. The rest of the signing process is similar to a typical digital signature process.

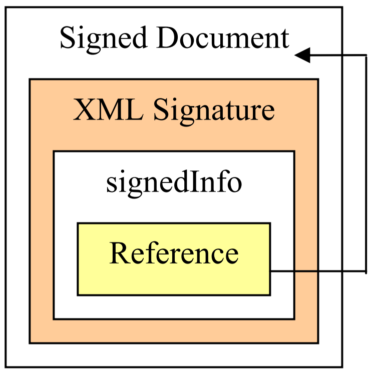
9.0 XML Signature Examples

XML signatures can be applied to any arbitrary data content. (e.g. any binary file such as a GIF image). Besides varying the data type, classification of XML signatures are as follows [4]:

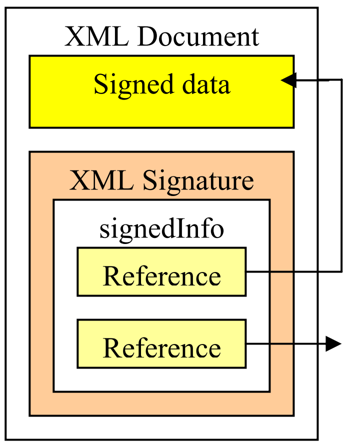
1. Those that are inside the XML Signature element are termed ‘enveloping signatures’ (refer to diagram below)



2. Those that are outside of, but surrounding the signature element is called ‘enveloped signature’ (refer to diagram below)



3. Those that are outside and disjoint from the signature element (the data is external to the signature element) are termed ‘detached’ signatures (refer to diagram below, a detached signature over two data items)



Because the signature elements is able to sign multiple pieces of data, there can be any two, or all three types of signatures at the same time (i.e. enveloping, enveloped or detached).

The code example 1 below shows an instance of a 'detached' signature [36]. (These series of examples were taken from the W3C signature recommendation document)

```
[01]    <Signature Id="MyFirstSignature"
        xmlns="http://www.w3.org/2000/09/xmldsig#">
[02]    <signedInfo>
[03]    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
        20010315"/>
[04]    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
[05]    <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
[06]    <Transforms>
[07]    <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
[08]    </Transforms>
[09]    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[10]    <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
[11]    </Reference>
[12]    </SignedInfo>
[13]    <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
[14]    <KeyInfo>
[15a]   <KeyValue>
[15b]   <DSAKeyValue>
[15c]   <p>...</p><Q>...</Q><G>...</G><Y>...</Y>
[15d]   </DSAKeyValue>
[15e]   </KeyValue>
[16]    </KeyInfo>
[17]    </Signature>
```

Example 1: An example of a simple detached signature

Comments to code example 1:

- The information that is signed is the "signedInfo" element (Lines 02 to 12)
- Reference to the algorithms used in calculating the "SignatureValue" element is shown within the signed section, while that element itself is outside the signed section (Line 13)
- The "SignatureMethod" reference (Line 04) is to the algorithm used to convert the canonicalized "SignedInfo" into the "SignatureValue". It is a combination of a key-dependent algorithm and a digest algorithm, i.e. DSA and SHA-1, with other manipulation such as padding
- The "KeyInfo" element (Lines 14 to 16, this element is optional) indicates the key used to validate the signature

10.0 Transforms

There is a wide range of possibilities to the order in which encryption, signing, and modifying can take place. For example, a user may need to enter data into a document that is already partially signed (or partially encrypted). And he may need to do this without preventing subsequent validation (or decryption).

The code example 2 below shows how a recipient is informed on the correct order of decryption and signature verification [4]. Note that the example shows the part of the document to be signed. Within the "order" element, the personal and financial details of the "cardinfo" element (lines 7 to 11), are in clear text but some encrypted data is already present (line 12).

```
[01]    <order Id="order">
[02]    <item>
[03]    <title>XML and Java</title>
[04]    <price>100.0</price>
[05]    <quantity>1</quantity>
[06]    </item>
[07]    <cardinfo>
[08]    <name>Your Name</name>
[09]    <expiration>04/2002</expiration>
[10]    <number>5283 8304 6232 0010</number>
[11]    </cardinfo>
[12]    <EncryptedData Id="enc1"xmlns="http://www.w3.org/2001/04/xmlenc#">...
        </EncryptedData>
[13]    </order>
```

Example 2: "Order" element within an XML document

The code example 3 below shows the "Signature" element (lines 1 to 26) includes the previous "order" element (lines 16 to 24), with its earlier plain text "cardinfo" element encrypted (line 22)

There are two transform references:

- decryption (lines 6 to 8) and
- canonicalization (line 9)

The decryption transform instructs the signature verifier to decrypt all the encrypted data except for the one specified on line 7 in the "DataRef" element. After the "EncryptedData" element (line 22) is decrypted, the "order" element is canonicalized, and the signature duly verified.

```
[01]    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
[02]    <SignedInfo>
[03]    ...
[04]    <Reference URI="#order">
[05]    <Transforms>
[06]    <Transform Algorithm="http://www.w3.org/2001/04/xmlenc#decryption">
[07]    <DataReference URI="#enc1" xmlns="http://www.w3.org/2001/04/xmlenc#"/>
[08]    </Transform>
[09]    <Transform Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-20001026"/>
[10]    </Transforms>
[11]    ...
[12]    </Reference>
[13]    </SignedInfo>
[14]    <SignatureValue>...</SignatureValue>
[15]    <Object>
[16]    <order Id="order">
```

```
<item>
<title>XML and Java</title>
<price>100.0</price>
<quantity>1</quantity>
</item>
<EncryptedData Id="enc2"
xmlns="http://www.w3.org/2001/04/xmlenc#">...</EncryptedData>
<EncryptedData Id="enc1"
xmlns="http://www.w3.org/2001/04/xmlenc#">...</EncryptedData>
</order>
</Object>
</Signature>
```

Example 3: "Order" document after signing and further encrypting and now showing transform information

11.0 The Validation Process

Signature validation requires the data object that was signed to be accessible. The signature will indicate the location of the original signed object. This can be referenced by a URI within the XML signature that is [24]:

- embedded within the XML signature (the signature is the "parent");
- reside within the same resource as the XML signature (the signature is a "sibling");
- has its XML signature embedded within itself (the signature is the "child).

This validation process is broken into three parts. XML Signature specifies that all these steps must be fulfilled before the signature is considered valid. The three parts are:

1. The signer's public key must be obtained. This key can be obtained by using the key information provided by the XML signature. In some cases, the public key may already be known.
2. Signature validation. The signature value can be checked by processing the "SignInfo" element according to the stated canonicalization and hash algorithms to obtain a hash value, and comparing the hash against the signature using the public key (as in normal signature verification).
3. Reference validation. The URI references to be checked for the correct digest value.

Verification failure would result if one of the references fails to provide the correct digest. If a signer does not wish to make reference validation compulsory, the references can be put in a Manifest element (enveloping signature).

12.0 XML Digital Signature Recommendation

The W3C's has officially recommended a new XML digital signature specification standard. The signature will be used to sign XML documents to enable recipients to verify the identity of the sender and the integrity of the data. It is formally known as XML-Signature Syntax and Processing

The recommendation, while built in XML and designed for XML documents, can be used to sign other kinds of documents [36].

13.0 Conclusion

XML digital signature is a critical foundation on top of which users can build more secure applications. By offering basic data integrity and authentication tools, XML digital signatures provide new functionalities for applications that enable different types of trusted transactions.

Chapter 7

XML Digital Signature Considerations

XML digital signature is specially designed to meet the needs of XML-enabled documents. This chapter looks at the various impacts and implication of XML digital signatures when it is used in the various applications

1.0 Introduction

Like non-XML digital signatures (e.g. PKCS), XML digital signatures adds data integrity, authentication and support for non-repudiation to the data that they sign. But unlike non-XML digital signatures, a XML digital signature is designed to both account for, and take advantage of the features of XML.

Regardless of whether it is XML-enabled or otherwise, a digital signature although hard to forge, is not foolproof. For example, the private key can be stolen, or an attack can be made on the digital signature protocols by changing the signed data. Changing the data that is signed would invalidate the signature. Hence, due considerations should be given when using them.

2.0 Implications of Security in XML-enabled Documents

Accompanying XML's many advantages and features are critical security issues. For example, as XML is primarily used in Internet-based communication, this provides the opportunity for others to sniff or spoof data. While XML allows business data to be more efficiently shared between multiple parties such as suppliers, customers, and governmental regulators, security breaches with such data can have very adverse consequences to the user.

A simple comparison between XML data and conventional data is as follows:

XML DATA	CONVENTIONAL DATA
Open standard	Closed standard
Self describing	Context dependent
Loose coupling	Tight coupling
Asynchronous	Synchronous
Decentralized	Centralized
Heterogeneous environment	Homogenous environment

From a security standpoint, a specification that is "self-describing", has "loose coupling", etc. would raise questions on whether the data could be secured. Thus, ensuring the security of XML data can be very complex [4], [26]. For example, when designing the flow of data within applications, considerations must be made to the four primary security objectives of confidentiality, integrity, authentication, and non-repudiation. Due to the flexible and ephemeral nature of XML, the user must establish and apply these objectives carefully.

As described in the earlier chapter, XML security mechanisms play a major role in ensuring secured systems. By using a data driven approach to security, XML entails more than just having, say, physical firewalls. For example, XML security mechanisms are able to specify exactly **who** gets to see **what**. It enables a single item of content to contain data elements of varying levels of sensitivity, and access permissions.

XML security mechanisms must extent across the user's entire network. Because of the presence of extranets, the user's data extends far beyond the user's physical control [2]. Hence, the user's security must be able to reach out as far as its data. XML's open system approach will allow for a company-wide security implementation that is manageable.

3.0 Impact on the architecture of the Internet

As new XML security mechanisms are put into place, new security architectures for the Internet might be required [2]. For example:

- The authentication by Web services for access control to other Web services
- In e-commerce payments service, authorisation tokens enabling access. The communication between customer and payment service could be in XML, and the tokens in XML format, signed by the payments service
- Website providing valuable information could have controlled permissions for publishing and accessing
- Key revocation and registration issues can be dealt with using separate services.
- Components of existing PKI functionality can be offered as Web services. New services to support a PKI could emerge, such as attribute certificates and authorisation servers.

As web services are still largely untested today, only time will tell if such ideas will happen.

4.0 Implications on Transforms

The transforms mechanism makes it easy to sign data derived from processing the content of an identified resource. For example, when signing forms, an XML application might permit users to enter certain data fields without invalidating a previous signature. The application could use a Transform (e.g. XPath-based) to exclude those portions of data fields that a user would change [4]. For this application, the things to be noted are:

- Only what is signed is considered secure as Signatures over a transformed document do not secure any information discarded by the transforms. Only what is signed is secure. Critical to signing XML is the use of canonicalization to ensure that all internal entities and XML namespaces are expanded within the data being signed. For example, all entities are replaced with their definitions. Hence, information that is not signed, but is merely part of an envelope containing signed information, is not secure. For example, unsigned recipient headers accompanying signed information within an encrypted envelope do not have their authenticity or integrity protected.

- Only what can be "seen" should be signed. If the intention of signing is to convey the consent of a user, then it would be necessary to secure, as closely as possible, the data as presented to that user. XML applications could achieve this by signing what was shown to the user (e.g. the screen image). However, the result would be difficult for subsequent manipulation. If the data is viewed though a browser, it would be reasonable to secure the browser version. However, it would not be practical, nor worthwhile to sign and preserve a complete copy of the browser and operating system that the browser was using.

- The signer should "see" what is signed. In situations where a user is 'seeing' and consenting (or approving) something, the user should sign only what he is able to 'see'. Automated mechanisms that trust a transformed document on the basis of a valid signature should operate over the data that was transformed and signed, and not the original, pre-transformation data.

For applications that operate over the original or intermediary data, potential weaknesses can occur between the original and transformed data. For example, consider a canonicalization algorithm that normalises character case from uppercase to lowercase. By changing the character case, a malicious user could introduce changes that are normalised and therefore not consequential to signature validity. Hence, the user could for example, influence the result of a query [23]. There will be a risk if the change is normalised for signature validation but the processor operates over the original data, and returns a different result than intended.

5.0 Implications on Security Assertion

The use of other XML standards could give XML signatures specific meaning. For example, in the Security Assertion markup language, the XML digital signature provides the authorisation for restricted actions.

As the Signature forms part of a security assertion, the technological aspects of digital signatures needs to be considered. For example, a signature validation requires:

(1) the retrieval of key information
(2) the checking of the revocation lists and
(3) verifying the signature

The reason for checking the revocation list is to negate compromised private keys. However, where security assertions are long term, a revocation mechanism for these assertions needs to be part of the infrastructure. In other words, although the private key is still valid, the assertions made for it are not [24].

6.0 Implication on Multiple Layer Security

When determining a communication session between authenticating users, employing a security structure that relies solely of XML security mechanisms might not be efficient. This is because relying on XML Signature for signing all the messages passed could be cumbersome. End-to-end security using mechanisms like SSL, or IPSec, may be sufficient and efficient enough for such transactions. Therefore, XML security mechanisms should be used only where it has an advantage over existing protocols [4].

However, new XML security mechanisms may find particular niches of application. For example, XML authentication tokens could be used at the beginning of a session, or XML Signature could be used at the end of a session for binding both parties to an agreement.

7.0 Implication of Digital Signature Strength

Signature strength refers to the difficulty to forge a signature on new or modified data. Like most signatures, signature strength depends on all the links in the security chain [14]. It includes:

- The signature and digest algorithms used
- The strength of the key generation
- The key size
- The security of the key, certificate authentication, distribution and storage mechanisms
- The certificate chain validation policy for certificate-based systems
- The protection of cryptographic processing from hostile observation and tempering

Hence, the overall security of the system depends on the security and the integrity of its operating procedures, the personnel, the administrative enforcement of the procedures, etc. However, some of these factors are beyond the scope of XML Signatures.

8.0 Implications on Legislation

There could be legislative problems with XML digital signature standard [2]. For example:

- Referring to the detached signatures, how can one decide whether reference validation was successful at a certain time in the past?
- A party may deny responsibility by stating that the XML document to which the signature referred was never present at the URI stated by the signature. And consequently, the signature was never valid.
- If a signature was removed on a certain date, but remained in a cache, is the party still responsible for a signature validated after this date?
- With the use of external references in order to be accessed, a URI could require either a cookie or authentication. Hence, consider a situation when a decision is made basing on information from an external website, and a digital signature is used referencing that website to ensure that the decision is valid only if the information is not changed. This information is copy-righted, and therefore cannot be placed in the document itself. Consequently, a detached signature is the only valid method. However, if a cookie is needed to obtain access to the website, it will be difficult to create the digital signature, and allow the reference validation to work properly

9.0 Conclusion

The XML digital signature is a generic standard, covering most methods by which a digital signature in XML format could be employed. As the signature standard provides a very flexible mechanism, there could be many ways to misuse them. This could result in producing insecure or misleading results.

One way to prevent abuse is to narrowly define a digital signature used in a document (or protocol). For example, a protocol may require that:

- the signature to be enveloped only
- the allowed transformations to be restricted to a few cases
- key information is required in one particular format

It is expected that there will be major effort to create XML security services. Perhaps, eventually, security at the XML level will settle down to a number of core areas, with other security necessities remaining at other layers, e.g. SSL or IPSec.

Chapter 8

The Future of XML

The successful adoption of XML depends on many factors. From the viewpoint of XML as a useful technology for data handling, this chapter explores the issues of the so-called XML hype, its myths, implications and impact, risks and future challenges

A critical analysis of the dissertation is included in the chapter

1.0 Introduction

XML has been touted as the data exchange standard that will revolutionise businesses. Generating a lot of buzz in the technology world, XML is seen as the *du jour* problem solver, the missing piece in the interface puzzle that will among other things, reduce costs, increase speed, enable a host of new application models, etc.

XML is seen as the solution to the long-standing challenge of interfacing new and legacy systems, and was the ideal solution for an age-old problem, interoperability. Hence, by adopting XML, users can ignore the problems of incompatible standards, devices, and formats. It was as if XML alone could act as a universal translator, a future-proofer.

But some of the basic truths of XML are [34]:

- XML is not a format. It is only a way of making formats. It sets rules for making sets of rules
- Using XML does not guarantee well-designed results. For example, if a XML user designs a particular format, the format is no more automatically interoperable than are two languages that use the same alphabets. Just as not all words in a sentence made from similar alphabets belong to same language (e.g. English and French), not all XML standards work together either.
- XML is no more resistant to the "garbage-in, garbage-out" syndrome than any other technologies

Therefore, the vision of XML as a pain-free method of describing and working with data is far from the perceived truth [20], [28]. For example:

- Designing a good XML format takes a lot of effort as it requires a rigorous description of the problems. Using XML does not remove the pain of having to describe the data upfront. The payoff comes only if it is rolled out carefully enough. For example, if implemented thoughtlessly, many problems will arise as a result of an improperly described data
- Interoperability is not an engineering issue. It is a business issue. Even if a XML technical standard is complete, the business standard is not. Because these standards have become important competitive tools, incompatibility still exists because of the reluctance of companies not wanting to cooperate with one another. As a result, defining unique XML tags might lead to compatibility problems later.

2.0 Implications of being Application Independent

One benefit of having common structures and an open standard is that it enables low degrees of application coupling.

Programming languages like Java has enabled portable code for cross-platform capability. Having portable data was seen as the next step for application-independent capability. But having data portability is a greater challenge than having portable codes. For data to become portable, (1) separating content from presentation and (2) defining specific vocabularies are required [26]. The vocabularies enable the data transfer between applications without loss of meaning.

Although the advantages of data portability are obvious, enabling it is problematic. One reason is due to the sheer size of existing non-XML data. Another reason is the diversity of businesses. This great diversity means that many different vocabularies need to be developed. This problem can be solved only if industries and application vendors mutually define and agree on these specific vocabularies [25]. This represents one of the biggest problems towards having data portability. If data from different sources adhere to common structures, applications can be built to consolidate the data and present it in a unified and portable way.

3.0 Implications on enabling Systems Integration

The use of commonly structured information is a key enabler for systems integration. However, this does not solve nor address all the technical issues as different systems integrate structured information differently. Hence, having structured information is only one part of the solution.

As mentioned earlier, the other part of the solution is in having standard vocabularies that must be defined, agreed upon, and used. For example, just because Airline “X” defines and uses the Air Travel Markup Language for flight information, it does not mean Airline “Y” will use it. And Airline “Z” might not know that ATML exists.

While XML is a standard, each XML vendor can interpret how to define. At best, this might result in application inconsistencies; and at worst, the creation of many different proprietary vocabularies [21]. Hence, one cannot assume that by using XML, it will completely alleviate dependencies on different vendors. It may simplify things. Conversely, it may introduce new interoperability problems.

4.0 Implications for HTML

XML adoption does not signal the death of HTML. Although comparatively, HTML is inflexible, it has served many needs. Furthermore, there may be little commercial benefit in converting existing HTML data systems to XML, as it will take a lot of effort and money [28]. By reasons of its sheer size, it is envisaged that HTML will be used for many more years, perhaps with a focus on document-centric applications, for documents that are intended for human consumption (as opposed to machine consumption)

But because XML enables users to do tasks that users could not do with HTML, XML might gain dominance over time [5]. Therefore, for potential XML user, it would be a safe strategy to consider using XML for new applications, or for major upgrades.

5.0 Implications of Metadata

Business data are valuable, critical assets. Unfortunately, in most legacy systems today, these data are stored in data repositories that does not support easy access or retrieval, nor allow easy cross-platform data exchange.

One way to improve data access is by the use of metadata. Well-defined metadata gives a user a competitive advantage as it enables a more seamless data sharing and transfers [28]. But the key for successful implementation is to have carefully defined, and universally accepted metadata.

Although the metadata concept is easy to understand, they are difficult to manage. And it is not easy to define them at consistent levels of detail. For instance a metadata may be used to:

- describe a collection of documents; or
- describe data elements within a specific document.

As users can create their own tags, they may not all tag their data in exactly the same way. This might result in having XML data requiring lots of effort to aggregate.

Access to XML data is dependent on the ability to interpret its structure. Since users create their own semantics, the key to unlocking these XML documents is to publish them (as DTD files or schemas) [5], [6]. But even with DTDs or schemas, the data might not be readily accessible without published standards, secure access, and applications to process and translate them.

6.0 Implications of an Open Structure

XML is said to enable universal access to data. For example, the recipe vocabulary example in Chapter 2 has a vocabulary structure consisting of Name, Portions, Ingredients, and Method. The idea is if all recipes are structured similarly, others can just parse that recipe data, and automatically enter the data into, say, a recipe database.

Although this idea is technologically good, there is one inherent problem. Owners of valuable or sensitive content would not want others to be able to automatically take their content. Publishers of copyrighted web content will be deprived of their advertisement revenue if anyone could just take their contents, minus say, the advertisements that originally came with the contents [25].

So why would content owners want their content to be well structured externally? Why would companies share strategic data with partners and competitors? In such cases, there are no incentives for making data so easily exportable. And if website owners resist XML, there would be no incentive for browsers to support it.

If data is difficult to extract, but easy to search for, there could be a stronger business case. In industries where there is little competition, there could be a good reason for a common vocabulary because it could mean that companies can sell more, efficiently. But in industries where there is great competition, companies will have no incentive at all to co-operate as it will lead to even more competition [6].

7.0 Impact on Databases

As with most systems, there is usually a trade-off when using XML databases. While XML databases enable users to extract XML data without using special formatting statements, the loss comes in the transacting logging and overall scalability of the system. Most XML databases today lack the mature transaction logging of larger more scalable systems (e.g. Oracle, SQL Server). As a result, it becomes slower as the number of records grows.

Modern databases systems such as Server 2000 have included XML parsers that allow users to easily create XML datasets. However, older database systems do not have this capability. Hence, to get the data out of these older databases, third party software would be required.

To import or export data, XML database systems would need an XML parser or translator. For XML data import, a translate document must be created to tell the system what fields exist and, if it is a relational database, what other tables exist and how these tables are related [7]. These translators may be in the form of a style sheet (e.g. XSLT) or a schema definition.

Although XML database implementations are typically fast, applications requiring large amounts of data to be processed (e.g. assembled, searched, sorted) to complex user rules are better implemented using traditional database systems (e.g. SQL Server). While XML may act as an intermediary to a database, it is not a replacement. Hence, existing traditional database applications are by no means obsolete. XML should not be used in situations where the data can be more effectively handled using existing technologies or infrastructure.

Furthermore, the use of extra tags in XML databases could increase the file size significantly. Typically, to about two to three times the normal size of a flat file. This is because of the tags surrounding each data element [25]. To solve this problem, data should not be stored with the tags.

8.0 Impact on Skills Base

Like other technologies, XML has an impact on the existing skill base of developers. Although the generic syntax of XML is similar to HTML, there are many concepts associated with XML that are not familiar.

XML does not necessarily simplify things. For example, while it successfully separates code from presentation, most implementations of XSL (the language that describes how to present the XML data) are just as complex as other programming languages.

Hence, developers will have to acquire new skills and new ways of approaching problems [35]. They must understand the capabilities, the various implications and issues associated with using XML.

9.0 Impact on Infrastructure

XML is designed for use over a public network, instead of proprietary networks. Hence, users need not have to invest resources in building direct links with multiple trading exchanges.

XML standards are implemented in software. But as they operate in a networked environment, they inevitably have an effect on the infrastructure. For example, using XML can either reduce or increase network traffic by the following:

- The use of large XML document, containing extensive data, can increase network traffic
- But, the increased use of local, client-side, processing can reduce network traffic

For the applications infrastructure, XML enables integration of disparate systems as data exchange between systems is made possible by using a common “language” platform [10].

10.0 Challenges of XML Adoption

Since its introduction, few technologies have been hyped as much as XML. While XML is seen by many as a key enabler to many things, it is really only the first step. For example, XML tags are supposed to provide a simple data format. But the intelligent defining of these tags, plus the common adherence to their standard usage will determine the real long-term value of XML. Potential XML users must not assume that XML to be a cure-all [35]. They should carefully analyze their requirements in order to apply XML appropriately.

One of the most visible drawbacks to XML is the requirement of a wide variety of standard vocabularies (DTDs and schemas). Already, there are several different XML vocabularies promoted by different industry coalitions or bodies like the W3C, and OASIS (Organization for the Advancement of Structured Information Standards). Hundreds of XML standards have been created since the introduction of XML [21], [36].

Furthermore, the issue of vocabulary convergence among different industries appears difficult to achieve. The effort to converge should not occur in a vacuum. For example, if a company tries to develop its own applications and DTDs, it may find that its development is incompatible with other companies [28]. Hence, collaboration with other parties is the key to success.

XML technical standards are currently in a state of flux. The creation of many different proprietary technical standards can undermine the successful adoption of XML. The flood of specification development endangers the discovery of solid XML standards.

Companies looking to use XML to gain a competitive edge must consider carefully parameters like process formats, integration interfaces, and business semantics. Compromises should be made to ensure widespread interoperability [10]. However, such trade-offs may make XML documents larger and less focused, but the increase in data sharing will be more than offset the compromise.

XML is a simple idea with powerful implications. But, it is no longer the pure, exciting force for openness we believed it to be. The potential for commercial success and the subsequent profits to be made has changed the nature of the XML industry to one that is quite political in nature. Hence, no one quite knows exactly how XML adoption will shape up eventually.

11.0 Critical Appraisal

The dissertation has achieved its aim to investigate and examine XML's features and capabilities and its security considerations, implications and impact to XML-enabled documents study, albeit the final dissertation submission is slightly modified from the original proposal.

The original dissertation proposal was to focus on two main issues of XML. They are:

1. What is XML? Covering the evolution of the Internet and markup languages such as SGML and HTML to XML itself, including the features, benefits and applications of XML
2. XML enabled security mechanisms, covering
    i. XML Digital Signatures; and
    ii. XML Encryption

During the research work and the actual writing of the dissertation, I discovered that the coverage of the proposed dissertation topic is too wide to be contained within the set maximum limit of 20,000 words.

On hindsight, the biggest error was to attempt to cover too-wide a topic. A major rework was hence necessary, but the dilemma was whether to be cover XML is depth or in breadth. To cover XML in greater depth would mean limiting the range of topics, but to cover XML in wider breadth would necessitate skimping through the varied topics.

The final decision was to leave out the topic on XML Encryption altogether. The two reasons for this decision were that firstly, I felt that sufficient focus should be made to research on the need for security mechanisms for XML-enable documents as opposed to non-XML-enabled documents. Secondly, greater focus was made on XML Digital Signature as the XML Encryption effort is not as mature as that associated with the XML Digital Signature standards at the point of writing.

The second error was that I spend too long a time to complete the dissertation. The unfortunate result was that research work on XML topics done earlier became obsolete, and the information obtained became less impactful and somewhat less 'original' when reported later.

Hence, this dissertation can by no means considered fully completed, nor a perfect piece of work. However, within the constraints of time and the cap to the length of the dissertation, I feel that the original objective of this dissertation has to a large extend, been met.

APPENDIX A

XML HISTORY

	EVENT
1969	Generalized Markup Language (GML), Charles Goldfarb, et. Al., IBM
1974	Standard Generalized Markup Language (SGML), Charles Goldfarb, et. al., IBM
1986	SGML adopted as a standard by ISO
1989	HTTP + HTML = World Wide Web, Tim Berners Lee, CERN
1996	Work commences on XML at World Wide Web Consortium (W3C)
1998	XML 1.0 specification released (W3C)
Present	On-going proliferation of XML as a major data representation language for the WWW; adoption by various sectors of industry and academia

APPENDIX B

XML STANDARDS

XML should not be viewed as a single technology as it is a collection of technologies surrounding a common core. The XML family of standards is managed by the W3C.

Some of the XML standards are:

- XML 1.0 - The base specification
- XML Namespaces – A means of identifying and separating XML vocabularies
- XSL (Extensible Style Language) – Includes XSLT for document transformation
- XPath, XLink, XPointer – Linking and addressing languages
- XML Schemas and XML Query

APPENDIX C

THE XML LANDSCAPE

<u>XML.ORG</u>

I N D U S T R Y S O L U T I O N S					
XMLife	S12	OTA	FpML	HL7	….

C R O S S - I N D U S T R Y S O L U T I O N S				
OBI, OTP, …	OAG	TBD	XML/EDI, RosettaNet	….

<u>OMG, IETF, etc.</u>

S O F T W A R E P R O D U C T S B u i l d R u n M a n a g e			
HTML, JSP, EJB, SVG, …	SMTP, POP3, MIME, …..	SQL, JDBC, ….	WebDAV, XMI, ….

<u>W3C</u>

C O R E   T E C H N O L O G Y
TCP/IP, HTTP, Java, XML, ….

B i b l i o g r a p h y

(The page numbers, etc., listed at the end of references indicate the parts of the reference actually used in the preparation of this dissertation)

[1] K. Ahmed, et al., *XML Meta Data*, Wrox Press Ltd, 2001, ISBN 1861004516, pp. 6-24, 32-57.

[2] B. Dournaee, *XML Security*, McGraw-Hill, USA, 2002, ISBN 0-07-219399-9, pp. 32-49, 107-146, 148-192.

[3] B. DuCharme, *XML: The Annotated Specification*, Prentice Hall, USA, 1999, ISBN 0-13-082676-6, pp. 3-63, 65-84.

[4] D.E. Eastlake, and K. Niles, *Secure XML, The New Syntax for Signatures and Encryption*, Addison-Wesley, USA, 2003, ISBN 0-201-75605-6, pp. 4-11, 35-48, 207-233, 248-252

[5] E. R. Harold, *XML Bible*, IDG Books Worldwide Inc, USA, 1999, ISBN 0-7645-3236-7, pp. 3-6, 80, 83, 134-160, 192-246.

[6] C. Horak, *The XML Shockware*, Software AG, Germany, 2000, pp. 3-17.

[7] IGNIA LLC., “Behind the Hype: XML”, *IGNIA LLC*, USA, 2000, sections “Benefits’, and “Tradeoffs”.

[8] F. Jung, *XML Backgrounder Technology and Applications,* Software AG, Germany, 2000, pp. 4-8.

[9] K. Kanakamedala, J. King, and G. Ramsdell, “The truth about XML”, *The McKinsey Quarterly* Number 3, USA, 2003, para 4-8.

[10] G. C. Kendell, “Three Myths of XML”, *XML.com*, O’Reilly & Associates, Inc., USA, 2001, sections “XML is Open and Free”, “Schemas are Magical”, and “XML is the Dog, Not the Tail”.

[11] A. Kotok, “Extensible and More”, *XML.com*, O’Reilly & Associates, Inc. USA, 2000, sections “Frameworks”, “Functions” and “Verticals”.

[12] A. Kotok, “Making XML Work in Business”, *XML.com*, O’Reilly & Associates, Inc. USA, 2000, sections “Information Management”, and “The Early Payoff”.

[13] M. Leventhal, D. Lewis, et al., *Designing XML Internet Applications*, Prentice Hall, USA, 1998, ISBN 0-13-616822-1, pp. 5-49, 51-79, 81-119.

[14] P. Lindstrom, “Special Report: The Language of XML Security”, *Network Magazine*, USA, 2001, section “XML Signature”, “XML for Security Functions”, and “The Power of XML”.

[15] M. Mactaggart, “An Introduction to XML Encryption and XML Signature”, *IBM developerWorks*, 2001, para. 2-3, 5-8.

[16] D. Martin, M. Birbeck, M. Kay, Michael; et al., *Professional XML*, Wrox Press Ltd, USA, 2000, ISBN 1-861003-11-0, pp. 9-11, 13, 15, 16, 21, 25, 152, 155, 157.

[17] H. Maruyama, and Imamura, "Element-Wise XML Encryption", *IBM Research, Tokyo Research Laboratory*, Japan, 2000, para. "Requirements".

[18] S. McGrath, *XML by Example: Building E-Commerce Applications*, Prentice Hall. USA, 1998, ISBN 0-13-960162-7, pp. 6-28, 32-68, 75-86.

[19] D. Megginson, *Structuring XML Document*, Prentice Hall, USA, 1998, ISBN 0-13-642299-3, pp. 3-40, 120-142, 144-172.

[20] NEDARC, "What is XML, and Can It Live Up to its Hype?", *National EMSC Data Analysis Resource Centre*, USA, section "Some Disadvantages".

[21] Organization for the Advancement of Structured Information Standards (OASIS). http://www.oasis-open.org/

[22] A. Selkirk, "Using XML Security Mechanisms", *BT Technology Journal*, Vol 19, No. 3, UK, 2001, pp. 35-43.

[23] A. Selkirk, "XML and Security", *BT Technology Journal*, Vol 19, No. 3, UK, 2001, pp. 23-34.

[24] E. Simon, and P. Madsen, "An Introduction to XML Digital Signatures", *XML.com*, O'Reilly & Associates, Inc. USA, 2001, section "Introduction to XML Signatures", and "The Components of an XML Signature".

[25] S. H. Simon, *XML E-Commerce Solutions for Business and IT Managers*, McGraw-Hill, USA, 2001, ISBN 0-07-137188-5, pp. 1-21, 23-40, 42-62, 87-104, 172-184, 207-215.

[26] S. St. Laurent, *XML A Primer*, Hungry Minds, Inc. USA, 1998, ISBN 0-76-454777-1, chap. 1-4, 12.

[27] S. St. Laurent, "When XML Gets Ugly", *XML.com*, O'Reilly & Associates, Inc. USA, 2000, para. 4, 8-11.

[28] B. Swart, "XML – Hit or Hype?", *TDMWeb*, USA, 2003, para. 2-6.

[29] D. Tidwell, "Introduction to XML", *IBM developerWorks*, 1999, sections "What is XML?", "How Can I Use XML Today?", and "Applying XML".

[30] D. Tidwell, "The XML Security Suite: Increasing the Security of e-business", *IBM developerWorks*, 2000, para 2-3, "Overview of Web security".

[31] R. Turner, *The Essential Guide to XML Technologies*, Prentice Hall, USA, 2002 ISBN 0-13-065565-1, pp. 1-54, 66, 70-71.

[32] W3C, "Extensible Markup Language (XML) 1.0", (Second Edition), *W3C.Org Technical Reports, World Wide Web Journal*, O'Reilly & Associates, USA, 2000, section 1.1, W3 Recommendation XML 1.0

[33] N. Walsh, “What is XML?”, *XML.com*, O’Reilly & Associates, Inc. USA, 2000, sections “XML Development Goals”, and “How is XML Defined?”.

[34] S. Withers, “XML: Great Hope or Great Hype*?”, Technology and Business Publication, ZDNet*, Australia, 2001, sections “ Communication Difficulties”, “Security & Authentication”, “Bandwidth Issues”, and “Diverging Standards”.

[35] R. Worden, “XML E-Business Standards: Promises and Pitfalls”, *XML.com*, O’Reilly & Associates, Inc. USA, 2000, sections “The Primer”, “The Pitfall”, and “The Way Forward”.

[36] World Wide Web Consortium (W3C), http://www.w3c.org