The University of Sheffield

Department of Computer Science

Applying Dynamic Trust Based Access Control to Improve XML Databases Security.

Submitted for the degree of Doctor of Philosophy
(PhD Thesis)

Norah Saleh Farooqi

September 2013

Supervisor: Dr Siobhán North

ABSTRACT

XML (Extensible Mark-up Language) databases are an active research area. The topic of security in XML databases is important as it includes protecting sensitive data and providing a secure environment to users. Trust based access is an established technique in many fields, such as networks and distributed systems, but it has not previously been used for XML databases. In Trust Based Access Control, user privileges are calculated dynamically depending on the user's behaviour.

In this thesis, the novel idea of applying Trust Based Access Control (TBAC) for XML databases has been developed. This approach improves security and provides dynamic access control for XML databases. It manages access policy depending on users' trustworthiness and prevents unauthorised processes, malicious transactions, and misuse from both outsiders and insiders.

A practical Trust Based Access Control system for XML databases was evaluated. The dynamic access control has been tested from security, scalability, functionality, performance, and storage perspectives. The experimental results illustrate the flexibility of Trust Values and the scalability of the system with small to large XML databases and with various numbers of users. The results show that the main research idea of this study is worth pursuing and the system could be developed further.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text. This work has not been submitted for any other degree or professional qualification except as specified.

Norah Saleh Farooqi.

ACKNOWLEDGEMENTS

First, I thank ALLAH for the guidance to complete this research.

I wish to thank my supervisor Dr. North who always supports and encourages me. I am really gratefully for her time, effort, and advice; for everything she did for me in every step of this research. She is an excellent supervisor and a wonderful person.

I would like to thank my fantastic great parents for their love, support, and advice. My success in this research is for my dear father and mother who spent many hours praying for me.

Many thanks to my lovely sisters for their support and encouragement to finish this work. Thanks also to my friends for their assistance and support.

Finally, I would like to express my thanks to Umm Al Qura University who offered me this chance to complete my academic studies.

LIST OF ABBREVIATIONS

BTF	Bad Transaction Factor
BTFW	Bad Transaction Factor Weight
BTNum	Bad Transaction Number
DAC	Discretionary Access Control
DOM	Data Object Model
DTD	Document Type Definition
EF	Error Factor
EFW	Error Factor Weight
ENum	Error Number
ETV	Existing Trust Value
ETVW	Existing Trust Value Weight
MAC	Mandatory Access Control
RBAC	Role Based Access Control
RDBMS	Relational Database Management System
SAX	Simple API For XML
SGML	Standard Generalised Mark-up Language
SQL	Structured Query Language
TBAC	Trust Based Access Control
TV	Trust Value
XACML	Extensible Access Control Markup Language
XLog	XML Log File for Security Rather Than Recovery
XML	Extensible Mark-Up Language
XPath	Xml Path Language
XQuery	Xml Query Language
W3C	World Wide Web Consortium

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Introduction

XML databases are an active research topic (Abiteboul et al., 2000; Champion, 2001; Tidwell, 2002; Oqbuji, 2004b; Vakali et al., 2005; Anderson, 2008; Jonge, 2008; Whatley, 2009; W3C, 2010; Palani, 2011; Sun and Wang, 2011; Abd El-Aziz and Kannan, 2012b; Abd El-Aziz and Kannan, 2012c; Noaman and Almansour, 2012; Verma et al., 2012; Desai, 2013; Thimma et al., 2013; Vela et al., 2013; W3Schools, 2013a). As with any database, they can contain sensitive and important data; therefore it is imperative to be able to provide a secure environment to deal with the data. This thesis concerns controlling access to data in XML databases.

Secure systems need access control to manage access to the data and prevent malicious processes. Traditional access models are limited in that they are static and focus mostly on protection from outsiders. The research described here is an attempt to address these limitations in the context of XML databases. The insider threat is a huge topic in data security and many methods have been proposed to identify misuse behaviour (Yi and Panda, 2003; Chinchani et al., 2005; Chagarlamudi et al., 2009), yet there has been no work on dynamic updates to access privileges in relation to trust for XML databases.

Trust Based Access Control has become established in many applications such as networks. It uses a trust management system that automatically calculates users' trust values. The trust values are updated according to an evaluation of the user's history.

In this thesis, Trust Based Access Control has been applied to XML databases in order to provide dynamic access control and solve misuse problems from both outsiders and insiders. It relies on evaluation of users' history of errors and illegal transactions and makes automatic updates in users' privileges according to their behaviour. The research hypothesis, motivations, and contributions are fully described in Chapter 5.

This short Chapter gives an overview of the thesis and Section 1.2 outlines its structure. While working on this research, a number of papers were published: Section 1.3 lists the published work.

1.2 Thesis Structure

The thesis can be divided into three parts. The first part, Chapters 1-5, discusses the related background work and explains the research aims. The second part, which consists of Chapters 6-8, describes the system design and components. The experimental results and evaluation are covered in the third part, namely Chapters 9-12.

The thesis consists of the following Chapters:

Chapter 1: Introduction. This Chapter gives a brief introduction to this thesis. It shows the thesis structure and lists published work.

Chapter 2: XML Background. This Chapter covers the basic concepts in XML. It includes components, tree structure, schema, query, and parsing techniques of XML.

Chapter 3: Related Work on Security in XML Databases. This Chapter discusses the existing types of access control systems. It describes several access techniques currently applied to XML databases.

Chapter 4: Related Work on Trust Based Access Control. This Chapter shows the features of Trust Based Access Control and explains the main concepts. It describes several models that were designed based on this approach and explains the calculations of Trust Value (TV).

Chapter 5: The Research Hypothesis. This Chapter explains the research motivations, objectives, and contributions. It highlights the research hypothesis. Some of the contents of this Chapter are based on previously published papers (Farooqi and North, 2011a; Farooqi and North, 2011b) .

Chapter 6: Trust Based Access Control for XML Databases. This Chapter describes the main outlines of the system design. The system consists of the trust module and the access control module. It also explains the system processes, the policy file, and boundary management. This Chapter is based on previously published papers (Farooqi and North, 2011b; Farooqi and North, 2012b; Farooqi and North, 2013).

Chapter 7: The Trust Module. This Chapter focuses on one of the two main components of the system. It explains the trust calculations, defining errors and bad transaction rules, capturing misuse, and logging. Some of the contents of this Chapter were published in previous papers (Farooqi and North, 2011b; Farooqi and North, 2012b; Farooqi and North, 2012c; Farooqi and North, 2012d; Farooqi and North, 2013).

Chapter 8: The Access Control Module. This Chapter describes the other main component of the system, the one which is responsible for handling Trust Values for data and users. It shows the access decision process in both access permitted and access denied situations. This Chapter is based on the work published in (Farooqi and North, 2012a).

Chapter 9: The Experiments' Design. This Chapter highlights the designs of seven different experiments to test the system implementation. It

shows the objectives and setup for all the experiments. It also discusses the tools, platforms, real world data sets, benchmarks, and user sets.

Chapter 10: The Results and Analysis. This Chapter illustrates the results for the seven experiments whose design is discussed designed in Chapter 9. The results analyse the performance, scalability, and security in individual modules and the integration of the whole system. The comparative results are discussed and presented graphically. Some experimental results were published in previous papers (Farooqi and North, 2012a; Farooqi and North, 2012b; Farooqi and North, 2012d; Farooqi and North, 2013).

Chapter 11: The Evaluation. This Chapter evaluates the practical works, highlights the strengths, and addresses the limitations of this work.

Chapter 12: Conclusion and Future Work. This Chapter summarises the thesis' findings and suggests interesting points for future research.

1.3 Publications

Some of the contents of this thesis were published in the following:

1.3.1 Peer Reviewed Papers

[1] FAROOQI, N. & NORTH, S. 2013. Performance Evaluation of Trust Based Access Control for XML Databases. *In- press, The Journal of Internet Technology and Secured Transactions (JITST),* Volume 2**,** pp. 1-8.

[2] FAROOQI, N. & NORTH, S. 2012d. A Performance Evaluation of Logging in XML Databases Using an XLog File for Trust Based Access Control. *International Journal of Intelligent Computing Research (IJICR),* Volume 3**,** pp. 337-341.

[3] FAROOQI, N. & NORTH, S. 2012c. Logging in XML Databases: XLog File for Trust Based Access Control. *World Congress on Internet Security (WorldCIS-2012).* Ontario, Canada IEEE Xplore, pp. 174-175.

[4] FAROOQI, N. & NORTH, S. 2012b. Evaluation of Practical Trust Based Access Control for XML Databases. *The 7th International Conference for Internet Technology and Secured Transactions (ICITST).* London, UK.: IEEE Xplore, pp. 336-340.

[5] FAROOQI, N. & NORTH, S. 2012a. Evaluation of Access Process in Trust Based Access Control for XML Databases. *The 6th Saudi Scientific International Conference (SIC).* Brunel University, London, UK.

[6] FAROOQI, N. & NORTH, S. 2011b. Trust-Based Access Control for XML Databases. *The 6th International Conference for Internet Technology and Secured Transactions (ICITST-2011).* Abu Dhabi, UAE: IEEE Xplore, pp.764-765.

[7] FAROOQI, N. & NORTH, S. 2011a. Developing a Dynamic Trust-Based Access Control Model for XML Databases. University of Sheffield, UK.

1.3.2 Posters

[8] "Trust Based Access Control for XML Databases" in Computer Science department, The University of Sheffield, UK, during the research retreat event in May 2011.

[9] "A Practical Trust Based Access Control for XML Databases" in Computer Science department, The University of Sheffield, UK, during the research retreat event in May 2012.

1.4 Conclusion

This thesis focuses on developing dynamic Trust Based Access Control to improve security and prevent misuse. The research objectives and contributions are described later in Chapter 5, after a discussion of the related work in XML security. The next Chapter gives the general background of XML and explains its main concepts.

2 XML DATABASES' BACKGROUND

2.1 Introduction

The Extensible Mark-up Language (XML) has become widely used. XML is commonly used to store, transfer, present, and retrieve data in many applications (Abiteboul et al., 2000; Champion, 2001; Tidwell, 2002; Oqbuji, 2004b; Vakali et al., 2005; Anderson, 2008; Jonge, 2008; Whatley, 2009; W3C, 2010; Palani, 2011; W3Schools, 2013a). Due to the recent increase in the availability of XML databases, much research has been undertaken to improve their usefulness. They are a relatively new kind of database but, like traditional databases, they require storage strategies and query languages. However, some important areas have not been thoroughly investigated. One of these areas is security in XML databases. As mentioned in the previous Chapter, this thesis revolves around providing secure access to XML databases using Trust Based Access Control.

This Chapter aims to outline the underlying concepts. The XML environment is very wide. This Chapter gives a general background for XML. It explains XML's main concepts, syntax, and schema in Section 2.2, 2.3, and 2.4 respectively. It shows how an XML file can be structured as a tree in Section 2.5 and how to navigate it using XPath in Section 2.6. The next Chapter will explore background in security for XML databases.

2.2 XML Concepts

The Extensible Mark-up Language (XML) has become widely used for structured data representation (Abiteboul et al., 2000; Champion, 2001; Tidwell,

2002; Oqbuji, 2004b; Vakali et al., 2005; Anderson, 2008; Jonge, 2008; Whatley, 2009; W3C, 2010; Palani, 2011; TotalXML, 2011; Thimma et al., 2013; W3Schools, 2013a). It was derived from SGML in 1996 and recommended by W3C in 1998 (Tidwell, 2002; Oqbuji, 2004b; Whatley, 2009; W3C, 2010; TotalXML, 2011) but differs from HTML since it focuses on storing and transferring data rather than controlling its appearance. There are many advantages of using XML. It is a self-describing language that gives users the freedom to create their own tags. It is known for its flexibility due to this feature (Tidwell, 2002). It is a simple text based language and portable data format (Abiteboul et al., 2000; Harold and Means, 2002; Tidwell, 2002; Ray, 2003; Whatley, 2009; W3C, 2010; Palani, 2011; W3Schools, 2013a) and readable by most platforms, so it can be shared between different applications.

XML files have many components, including elements (e.g. <Customer>), attributes (e.g. birthdate="16-6-1987"), and comments (e.g. <!-- Written by NSF -- >) (Walsh, 1998; Abiteboul et al., 2000; Tidwell, 2002; Whatley, 2009; W3Schools, 2013a). Figure 2.1 shows the XML document for a bank database. The three main components of the XML file are summarised in the next Section.

2.3 XML Syntax

2.3.1 Elements

The XML element is considered the basic component of the file. Normally, it includes the opening tag e.g. <Customer> and a matching closing tag </Customer> but it can be an empty tag <Customer/>. The value of the element, that which is enclosed between the pair of tags, can be a text, other elements, or both. Elements can include attributes such as <Customer birthdate="16-6-1987"> (Walsh, 1998; Abiteboul et al., 2000; Tidwell, 2002; Whatley, 2009).

```
<?xml version="1.0"?>
<!- - Written by NSF- ->
<Bank>
<Customer birthdate=”16-6-1987”>
<Name> john smith </Name>
<Mobile> 07777777 </Mobile>
<Address> city name, street name, postcode </Address>
<Balance> £2000 </Balance>
<Card>
<Number> 192837465 </Number>
<Start date> 11-11-2010 </Start date>
<End date> 11-11-2012 </End date>
<Security code> 2222 </Security code>
</Card>
<Transactions>
<Transaction id=”3333”>
<Type> draw </Type>
<Amount>£80 </Amount>
</Transaction>
<Transaction id=”3334”>
<Type> credit card payment <Type>
<Amount>£40 </Amount>
</Transaction>
</Transaction>
</Customer>
<Customer birthdate=”7-8-1950”>
…
…
</Customer>
<Customer birthdate=”27-9-1970”>
…
…
</Customer>
</Bank>
```

Figure 2.1 The bank XML database

2.3.2 Attributes

XML attributes provide information about the element that is not usually changed. They are placed inside the opening tag and consist of the attributes name and its value. The attribute value is placed inside quotation marks. There are also reference attributes that are used as pointers to the element itself or to

other elements (Walsh, 1998; Abiteboul et al., 2000; Tidwell, 2002; Whatley, 2009).

Attributes should be distinguished from elements. Attributes are used for specific and static values. Dynamic and changeable data are normally included as elements. As mentioned above, the element value can be a string, other sub-elements, or both. Attributes are more difficult to use and maintain than elements, so it is useful to limit their use (Abiteboul et al., 2000; Tidwell, 2002; Ray, 2003; Whatley, 2009; W3Schools, 2013a).

2.3.3 Comments

XML is a simple and clear language but, as with any language, comments are used to clarify the complexity of the code or to add notes for the writer or reader. The comment syntax is exactly the same as in HTML (e.g. <!-- Written by NSF -- >) (Tizag.; Harold and Means, 2002; Tidwell, 2002; Ray, 2003; Whatley, 2009; W3Schools, 2013a).

2.3.4 Well-Formed File

Although XML is flexible and gives freedom to users to create their own tags, there are some basic rules that should be followed:

- An XML file should have one root element.
- Starting and ending tags must match and are case sensitive.
- Nested elements should be ordered; the most recent opening tag should be closed before closing any earlier opening tags.
- The value of attribute should be placed between quotation marks.
- An attribute name inside an element should be unique.

When the XML file conforms to all these rules, it is called well-formed XML (Abiteboul et al., 2000; Harold and Means, 2002; Ray, 2003; Whatley, 2009; W3Schools, 2013a).

2.4 Schemas

Schemas are a major topic in XML. However, this Section gives only a brief introduction to this topic because it is not relevant to much of this thesis except in a few specialist areas.

In general, a schema is a database description that is developed in the design stage of the databases and it is quite static (Molina et al., 2009; Elmasri and Navathe, 2011). In the context of XML, a schema is used to store the file structure and show the elements' relationships. Several types can be used with XML files, such as DTD (Abiteboul et al., 2000; Lee and Chu, 2000; Harold and Means, 2002; Chase, 2003b; Ray, 2003; Molina et al., 2009), XML Schema (Abiteboul et al., 2000; Lee and Chu, 2000; Radiya and Dixit, 2000; Harold and Means, 2002; Ray, 2003; Molina et al., 2009; Waldt, 2010), RELAX NG (Chase, 2003a; Ray, 2003), and Schematron (Lee and Chu, 2000; Ray, 2003; Oqbuji, 2004a). The most popular ones are DTD and XML Schema.

DTD is the oldest way to describe an XML file's structure. It lists all the contents: elements and attributes. It can be defined inside the XML file as internal DTD or outside in a separate file as external DTD. The external DTD is used more than the internal because it can be related to many XML files and can define their syntax. DTD limitations such as namespace problems and lack of data types were overcome by introduction of the XML Schema. XML Schema is more readable than a DTD. It can also be defined as an internal XML Schema or an external file. A well-formed XML file is called valid XML when it has DTD or schema (Abiteboul et al., 2000; Radiya and Dixit, 2000; Harold and Means, 2002; Tidwell, 2002; Chase, 2003b; Ray, 2003; Molina et al., 2009; Whatley, 2009; Waldt, 2010; Elmasri and Navathe, 2011).

2.5 Tree Structure

An XML document is usually represented as a tree that starts with a root element and branches to many sub-trees that end with leaf nodes. The tree has nodes that reflect XML file components such as elements and attributes (Abiteboul et al., 2000; Darugar, 2000; Harold and Means, 2002; Ray, 2003; W3Schools, 2013a). The tree idea is derived from graph theory. An XML file can have only one tree representation (Ray, 2003).

The tree shows the XML document components and reflects its structure in graphical form. Many relationships between nodes can be defined using the tree. The root element is called the parent and it has many children in the lower level. The children at the same level are called siblings (Abiteboul et al., 2000; Harold and Means, 2002; Ray, 2003; W3Schools, 2013a). All these relations are described further in the XPath Section (2.6.1). Although, the tree based structure provides easy random access to either data or structure it consumes a large amount of memory (Darugar, 2000). Figure 2.2 shows the XML tree for the XML document in Figure 2.1.

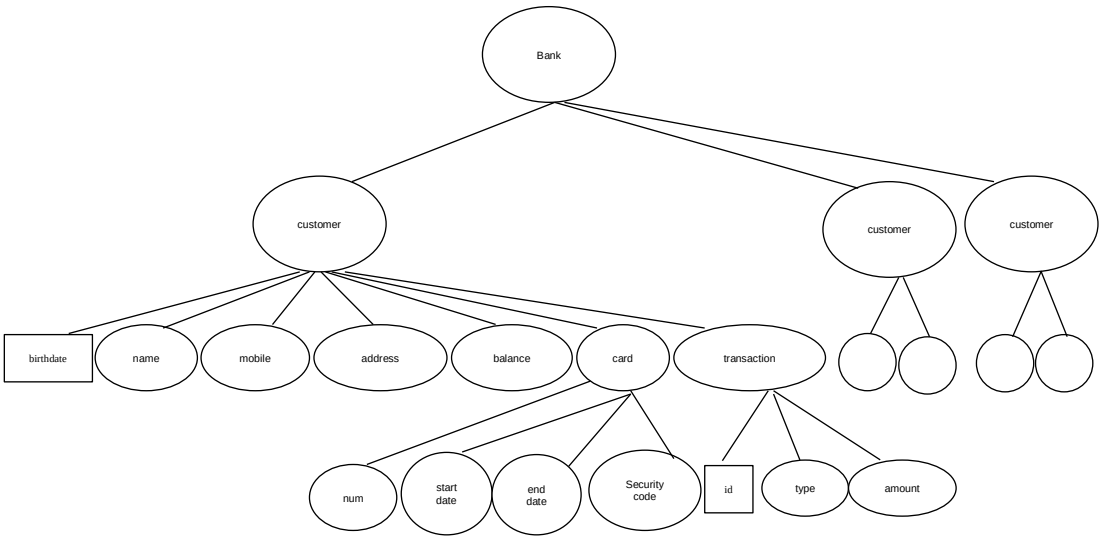


Figure 2.2 The tree of the bank XML database

2.6 Query Languages

XML files require a query language to extract data. In this Section, the two most popular XML query languages, XPath and XQuery, are discussed.

2.6.1 XPath

XPath is XML path language, which uses path expressions to navigate an XML tree and address specific parts, either a node or a set of nodes. The path expression is similar to that of file systems. XPath was recommended by W3C in November 1999. It handles the XML file as a tree. It is used as a simple query language and supports other query languages, such as XQuery. It defines nodes' relationships: parent, child, sibling, ancestor, and descendant. The parent and child relation is formed between adjacent levels of the tree. Each child has only one parent on a higher level. The parent can have any number of children including zero. Children at the same level with the same parent are called siblings. The ancestor relation goes up from any node until it reaches the root. In contrast, descendant goes down from a node until it reaches the leaves (Tizag.; Harold and Means, 2002; Ray, 2003; Molina et al., 2009; W3C, 2010; Elmasri and Navathe, 2011; W3Schools, 2013a).

XPath expressions show the location of elements and attributes in XML files. The syntax of XPath includes many special marks to identify the node (Tizag.; Harold and Means, 2002; Ray, 2003; Molina et al., 2009; W3C, 2010; Elmasri and Navathe, 2011; W3Schools, 2013a); the basic marks are explained in Figure 2.3 (W3Schools, 2013a).

Expression	Description
nodename	Selects all child nodes of the named node
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

Figure 2.3 Basic marks are used in XPath expression (W3Schools).

To explain how these marks are used, here are some examples of XPath expressions for the bank database, shown in Figure 2.1 on page 9:

/bank/customer/mobile

This expression can be used to access the mobile node that is a child of customer node and a descendant of bank node.

/bank/transaction@id

This expression is used to select the identifier (id) attribute of a transaction node.

Predicates can be used in XPath expressions to find specific nodes and values. They are inserted in square brackets "[ ]" (Tizag.; Harold and Means, 2002; Ray, 2003; Molina et al., 2009; W3C, 2010; Elmasri and Navathe, 2011; W3Schools, 2013a). For example, to find all transactions that have an amount over £50, the predicate is used in the XPath expression as follows:

/bank/customer/transaction [amount>50]

Furthermore, XPath expressions use axes to identify groups of nodes related to the specific node (Tizag.; Harold and Means, 2002; Ray, 2003; Molina et al., 2009; W3C, 2010; Elmasri and Navathe, 2011; W3Schools, 2013a). Figure 2.4 shows XPath axes' names and results (W3Schools, 2013a).

AxisName	Result
ancestor	Selects all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self	Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself
attribute	Selects all attributes of the current node
child	Selects all children of the current node
descendant	Selects all descendants (children, grandchildren, etc.) of the current node
descendant-or-self	Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself
following	Selects everything in the document after the closing tag of the current node
following-sibling	Selects all siblings after the current node
namespace	Selects all namespace nodes of the current node
parent	Selects the parent of the current node
preceding	Selects everything in the document that is before the start tag of the current node
preceding-sibling	Selects all siblings before the current node
self	Selects the current node

Figure 2.4 XPath axes (W3Schools)

XPath expressions can be classified into two types: absolute path and relative path. The absolute path for a specific node is the full path starting from root to the node. The relative path is shorter and is based on a specific node only (Tizag.; Harold and Means, 2002; Ray, 2003; W3C, 2010; W3Schools, 2013a).

2.6.2 XQuery

This Section only gives an overview of XQuery because it is not very relevant to this work. XQuery is a language used to query and extract data from XML files. It is based on XPath with extensions to cover XPath's limitations. XQuery provides the ability to handle queries using functions. It is flexible and can deal with complex queries. It provides FLOWER expression (FOR, LET, WHERE, and RETURN) to extract data, similar to SQL in relational databases (Cameron, 2008; Molina et al., 2009; Boag et al., 2011; Elmasri and Navathe, 2011; W3Schools, 2013b).

In general, XPath is considered a simple query language for XML files, and XQuery a more general and powerful one. Nevertheless this research will use XPath due to its simplicity and clarity and because the complexity of queries is not relevant. Using XPath in the system is discussed more fully in Chapter 9.

2.7 XML Parsing

The parser is an important component for processing an XML file. It is included in all applications that use XML. It aims to parse the XML text and create a representation as a tree or stream. Many parsers are used to construct XML files, such as DOM, SAX, JDOM, and Xerces2. Both DOM and SAX are discussed in the following Sections because they are the most popular and widely used.

2.7.1 DOM

Data Object Model (DOM) was developed by W3C (Hégaret et al., 2005). DOM parses an XML file and constructs it as a tree of objects in a memory. This tree represents the content of the XML file and shows the relationships between objects such as parent, child, and sibling. It converts each element in the XML file to a node in the tree (W3C, 2003; Eriksen, 2004) as described in Section 2.3.

DOM offers an easy method to navigate, access, and manipulate the XML data (Frank et al., 2003). It supports traversal in any direction and allows both read and write processes simultaneously. It provides random access to XML data using the tree structure (W2, 2008; W1, 2009). Using DOM offers a suitable environment for XPath (Berglund et al., 2010) and handling queries and updates (Al-Badawi, 2010). This parser can be used in several platforms including .NET, C++, and Java (Zhang, 2006).

However, creating and loading the XML tree into memory consumes storage space and run time. Large XML files face obstacles when parsed under DOM due to requiring a large memory space. Figure 2.5 shows how DOM represents the XML file as a tree.

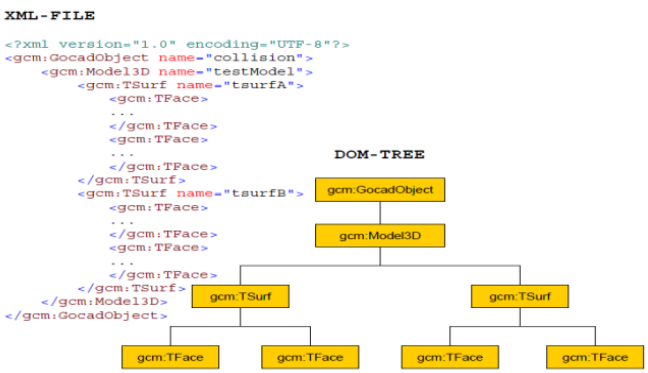


Figure 2.5 DOM tree for the XML file (Frank et al., 2003)

2.7.2 SAX

SAX is an acronym for Simple API for XML (Brownell, 2002; SAX, 2004). It parses the XML file and creates a stream based on events. Each event represents an element in the XML file. The order of events follows the order of elements in the XML file. It is simple, fast, and provides high level of performance in parsing because it does not store the XML file in the memory (Nazmul, 1999) and so supports the parsing of large XML files, unlike DOM.

However, it is limited to reading the XML data with no manipulation. It restricts navigation by providing only a top down traversal and a sequential access to data. Therefore, back navigation is not possible (W2, 2008; W1, 2009). In general, it focuses on parsing and creating events. Objects can be created by interrupting these events. Figure 2.6 shows events of the XML file that are created by SAX.

Figure 2.6 SAX events for the XML file (Frank et al., 2003)

Each of these two parsers has its advantages and disadvantages. The choice between DOM and SAX depends mainly on the system requirements. DOM was selected as the parser at the implementation stage in this research. This is because the system needs to know the whole structure of the XML file, uses random access to nodes, and does traversal in any direction. All these

requirements are provided by DOM but are not supported by SAX. However, using DOM in the implementation limits the scalability test when evaluating the huge XML databases due to storage restrictions. This limitation is discussed in Chapter 9.

2.8 XML Databases

An XML file can be either data-centric or document-centric. The data-centric type reflects that data in XML is highly structured and is commonly stored in databases. The document-centric type concerns semi structured textual content such as books (Bourret, 2005; Sun and Wang, 2011; Noaman and Almansour, 2012). Only the data centric type is relevant here due to its relationship to databases application.

There has been a debate as to whether XML is a database or not. XML can be considered a technology that used to build databases since it has the ability to store and retrieve data like other types of databases (Bourret, 2005; Sun and Wang, 2011; Noaman and Almansour, 2012). It includes many common databases features: it stores data in XML files, owns schemas (DTD and XML Schemas) and query languages (XPath and XQuery), and provides interfaces based on programming languages such as DOM and SAX. At the same time it lacks many features of database management systems such as security, multi-access, and recovery (Steegmans et al., 2004; Bourret, 2005; Noaman and Almansour, 2012). These limitations call into question XML's status as a database. Researchers have been concerned about these limitations and have tried to develop the XML database environment. This research is one such attempt. It aims to improve security in XML databases.

An XML database can be categorised into either an enabled XML database or a native XML database (Steegmans et al., 2004; Bourret, 2005; Molina et al., 2009; Papamarkos et al., 2009; Elmasri and Navathe, 2011). The enabled XML database stores data based on existing approaches using traditional

databases such as relational databases. The most important feature of using this type is to support existing applications, since a large number of XML files are already stored in relational databases (Steegmans et al., 2004; Papamarkos et al., 2009; Abd El-Aziz and Kannan, 2012b). This type depends on well-known and familiar approaches. It requires mapping techniques to transfer data from the XML structure to the relational structure (Steegmans et al., 2004; Elmasri and Navathe, 2011). It suffers from limitations. It does not handle large XML files well due to number of joins (Papamarkos et al., 2009). It is not concerned about hierarchical structure, nested data, and elements order. Some information may be lost during the conversion (Steegmans et al., 2004; Bourret, 2005; Sun and Wang, 2011; Noaman and Almansour, 2012).

The second approach is native XML databases, which are based on an XML file as the basic unit. This type is an appropriate approach to manage XML databases (Fiebig et al., 2002; Steegmans et al., 2004; Sun and Wang, 2011). It can easily be searched and its content managed because it is all in one place (Bourret, 2005; Sun and Wang, 2011). The native approach supports XML query languages, which improves the retrieval process (Steegmans et al., 2004; Bourret, 2005; Papamarkos et al., 2009; Sun and Wang, 2011). It is more flexible than XML-enabled databases (Bourret, 2005). The main limitation of this type is that it provides data in only XML format (Bourret, 2005; Abd El-Aziz and Kannan, 2012b). This approach can also be classified into two types according to Bourret (2005) and Papamarkos et al. (2009): text-based and model-based. The text-based approach handles the XML file as text and stores it as a file in the file systems or in relational databases as a CLOB/BLOB. The model-based type handles XML data as objects and the file is represented as a tree, as in DOM (Staken, 2001; Steegmans et al., 2004; Bourret, 2005; Harold, 2005; Sun and Wang, 2011; Noaman and Almansour, 2012). This research will focus only on native XML databases.

2.9 Conclusion

XML is a vast topic and not all of its aspects were covered in this limited Chapter. The basic points are included to give sufficient background to understand the research aims and the systems' platforms. The next Chapter is a literature review of access control systems since the topic of the thesis is access control for XML.

3 RELATED WORK ON SECURITY IN XML DATABASES

3.1 Introduction

XML databases are widely used in many different areas. Like any databases, they are used to store, retrieve, and provide data and information in an organised manner. They are multiuser systems, meaning they can be accessed by millions of users and they can provide a huge amount of data. This large amount of data needs to be controlled, managed, and organised. In addition, this data can be sensitive and personal. All data and especially confidential data need to be protected and saved in a secure environment. Therefore, XML databases should manage data securely to protect user rights and data privacy from loss or misuse (Izadi et al., 2007; Li and Hong, 2008; Gollmann, 2011; Thimma et al., 2013). This thesis focuses on the access control, which is one of the main techniques to improve security in XML databases.

In this Chapter, the general background of XML security is discussed in Section 3.2. The rest of the Chapter describes the work related to access control. In Section 3.3, access control concepts are explained and compared. Section 3.4 provides a literature review of several types of access control. In Section 3.5, access control techniques that are currently applied to XML databases are discussed in detail. Labelling technique is also described in Section 3.6 due to its relationship to access control.

3.2 XML Security

The main aim of XML security technologies is to protect information and ensure users have proper authorisation. These technologies include XML signature, XML encryption, and XML access control (Jo et al., 2005; Ardagna et al., 2007; Zhu et al., 2009; Myint, 2010; Gollmann, 2011). Both signature and encryption techniques are low level features whereas access control is considered a high level approach to security policies (Cho et al., 2002). Digital signature and encryption focus on making the data itself secure but access control provides secure access to data (Verma et al., 2012).

In general, all three techniques are used for data protection, but each one provides specific features that make it different from the others. Access control essentially concerns data confidentiality. Encryption is also used to preserve confidentiality when the data is transmitted via different platforms. Digital signature aims to prevent data tampering (Bertino and Sandhu, 2005). Security in XML includes communication security and managerial security. Both digital signature and encryption are designed to handle communication security, but access control approach works on the managerial security (Myint, 2010). W3C recommended XML signature, XML encryption, and XML key management specification (XKMS), which is used to support both signature and encryption (Ardagna et al., 2007; Ekelhart et al., 2008; Verma et al., 2012).

Some XML-based access control languages were developed by commercial companies. They are considered as specification security languages based on XML. XACML and SAML are products of the OASIS security services technical committee and XACL was proposed by IBM (OASIS; Hada and Kudo, 2000; OASIS, 2005). XACML is Extensible Access Control Mark-up Language that declares the access control policy and aims to provide a standard terminology among multiple systems. It is built on XML and is combined with the RBAC approach for XML (OASIS). SAML is Security Assertion Mark-up Language that is used for authentication and authorisations between identity and

service providers online. XACL is XML Access Control Language that specifies policies on subject, object, action, and condition to enforce security in XML documents. However, these languages have limited features that have been extended and improved (Bertino and Sandhu, 2005; Abd El-Aziz and Kannan, 2012a).

The access control technology is one of the main approaches to guarantee security and some authors describe it as the most effective one in XML databases (Hitchens and Varadharajan, 2001; Li et al., 2005; Qi et al., 2005; Lee and Yu, 2008; Li and Hong, 2008; Gonzalez et al., 2009; Zhu et al., 2009). It is also one of the main issues in XML (Gonzalez et al., 2009; Sun and Wang, 2011). At present there are no standards and well-defined rules for access control in XML (Lee and Yu, 2008; Gonzalez et al., 2009). This is clearly a point worthy of more investigation. This thesis focuses only on the access control technology. The work related to an access control approach is described in detail in the following sections.

3.3 Access Control Concepts

Access control is an important topic in security and it is applied in many computer fields such as operating systems, networks, and databases (Bertino and Sandhu, 2005; Chin and Older, 2011; Elmasri and Navathe, 2011; Gollmann, 2011; Hui et al., 2011). This research focuses on access control for XML databases. In general, access control models control and manage users' access to XML nodes or attributes according to policy rules. They help to prevent inappropriate access that breaches data security. The following Sections cover policy rules' concepts and the access control structure.

Any access control technique requires a security policy that defines the access levels for different users. It specifies who can access what and with what privileges. These rules are also called authorisation specifications (Gabillon, 2004; Qi et al., 2005; Di Vimercati et al., 2008). The term 'privileges', which is

frequently used in this area, means a right that is given to a subject to perform specific actions and operations on objects (Damiani et al., 2005; Gollmann, 2011). Usually the main syntax of policy rules is represented as <subject, object, action>. Subjects are entities that can access and achieve resources and data through requests. Each subject may be a user's identification, his location, IP address, or a combination of them; it can also be a group of users. Objects can be defined as items that are accessed through requests. Each Object can refer to an XML document, a single node, or an attribute. Action refers to privileges such as read (browse) and write (insert, update, delete); it is also called access mode (Chan et al., 2004; Gabillon, 2004; Di Vimercati et al., 2005; Gabillon, 2005; Jo et al., 2005; Di Vimercati et al., 2008; Li and Hong, 2008; Gollmann, 2011; Thimma et al., 2013).

Access authorisation is sometimes (Damiani et al., 2001; Di Vimercati et al., 2005; Jo et al., 2005; Di Vimercati et al., 2008; Byun and Park, 2010) also extended by adding sign and type factors to the main syntax and appears as <subject, object, action, sign, type>. The sign is represented by positive + or negative – marks. It reflects that the access is either permitted or denied. The type refers to the authorisation type, which is local, recursive, soft, hard, or a mixture of them. The term local means the authorisations apply for an element and its attributes only while recursive includes the element, sub elements, and their attributes. Soft refers to authorisations that are applicable for instance level which means restricted to a specific XML file. In contrast, hard means the authorisation is at schema level, which indicates that the subject can work with any XML file based on a DTD (Di Vimercati et al., 2008) . Li and Hong (2008) developed temporal access control policies by adding time factor to this syntax. These access policies can relate to read, write, and position privileges (Park et al., 2004; Gabillon, 2005; Jo et al., 2005; Qi et al., 2005; An and Park, 2007; Di Vimercati et al., 2008; Lee and Yu, 2008). The position privilege concerns

hiding the node content while showing that the node exists. This is described further in Section 3.5.1.

The main conceptual access control structure is an access control matrix that stores and manages access rights. An access control matrix contains a set of subjects, actions, and objects. Each subject is represented as a row and each object as a column in the matrix. Cells in the matrix may contain action values or be empty. The matrix consumes space due to empty cells and does not work efficiently with a large amount of data because the update process becomes difficult (Sandhu and Samarati, 1994; Chin and Older, 2011; Gollmann, 2011).

Usually, the access matrix is implemented using two traditional models: access control list and capabilities. An Access Control List (ACL) stores the access rights based on the object. Each object has a list that contains all subjects that can access it and the action values (Sandhu and Samarati, 1994; Qi et al., 2005; Lee and Yu, 2008; Chin and Older, 2011; Gollmann, 2011). In contrast, the capabilities model stores the access rights based on the subject. Each subject has a list of permitted objects with the action values (Sandhu and Samarati, 1994; Qi et al., 2005; Lee and Yu, 2008; Chin and Older, 2011; Gollmann, 2011). This type is related to Discretionary Access Control (DAC) (Gollmann, 2011) that is fully described in Section 3.4.1. Both techniques can be applied to simple structured systems. They store only the positive authorisations.

Although, the access matrix and its techniques are not relevant to this research, they were described above because they are fundamental points in the access control. In the following Section, the main access control categories are discussed.

3.4 Access Control Types

There are several classifications for access control models from different perspectives. Access control models can be categorised into simply: the

Discretionary Access Control model (DAC) and the Mandatory Access Control model (MAC) (Hitchens and Varadharajan, 2001; Zhu et al., 2007; Zhu et al., 2009). However, some authors categorise them into three core categories: Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role Based Access Control (RBAC) (Sandhu and Samarati, 1994; Jeong et al., 2003; Chan et al., 2004; Wang and Osborn, 2004; Zhang et al., 2004; Li et al., 2005; Zhang and Xue, 2005; Lee and Yu, 2008; Rashid et al., 2010; Thimma et al., 2013). In addition to these traditional categories, there are some new types: Attribute Based Access Control (Bobba et al., 2010; Kuhn et al., 2010; Junbeom and Dong Kun, 2011) and Trust Based Access Control (Bin and Shijin, 2010; Hua and Lili, 2010; Ma et al., 2010; Xing et al., 2010; Yang et al., 2010; Zhang and Rao, 2010; Singh, 2011). Likewise, there are other proposed types for access control such as function based access control (Qi et al., 2005), and purpose based access control (Byun et al., 2005; Sun et al., 2010; Sun and Wang, 2011). All these types are explained in the following Sections.

3.4.1 Discretionary Access Control Model (DAC)

Discretionary Access Control depends on the subject identity to manage access to databases. This type is characterised by its flexibility, so the subject can grant access control rights to other subjects (Sandhu and Samarati, 1994; Jeong et al., 2003; Chan et al., 2004; Bertino and Sandhu, 2005; Zhu et al., 2007; Zhu et al., 2009; Rashid et al., 2010). This flexibility leads to the implementation of Discretionary Access Control in several applications through Access Control List (ACL) (Rashid et al., 2010). It is adopted by the majority of commercial database management systems (DBMSs) (Sandhu and Samarati, 1994; Chan et al., 2004; Bertino and Sandhu, 2005; Rashid et al., 2010). However, the passing of access rights to other users may cause inappropriate access to sensitive information or permit malicious attacks on databases (Jeong et al., 2003; Rashid et al., 2010). Discretionary Access Control risks a loss of control of data and the ability to check prohibited flows of unauthorised subjects like Trojan horse

(Sandhu and Samarati, 1994; Zhu et al., 2007; Zhu et al., 2009; Rashid et al., 2010).

In the context of XML databases, many models that handle different approaches to query, update, view, and access data are based on DAC (Bertino et al., 2000; Damiani et al., 2001; Damiani et al., 2002; Di Vimercati et al., 2005; Murata et al., 2006; Damiani et al., 2007). Some of these models that are developed by Damiani et al. make the authorisation processes for accessing XML databases depend mainly on the DTD. It appears that the literature on DAC in XML databases used this basic type of access control because of its simplicity. These models were designed to investigate other points in XML databases and included DAC as a basic access control. So, the mechanism of applying DAC to XML Databases was not explained clearly or in detail.

3.4.2 Mandatory Access Control Model (MAC)

In Mandatory Accesses Control (MAC), both subjects and objects are classified into multiple-levels depending on the object's content. Security levels have labels and the security level of an object reflects its sensitivity (Sandhu and Samarati, 1994; Jeong et al., 2003; Chan et al., 2004; Bertino and Sandhu, 2005; Zhu et al., 2007; Rashid et al., 2010). These security levels can be ordered or unordered. The order levels are usually classified into TopSecret, Secret, Confidential, and Unclassified (Sandhu and Samarati, 1994; Jeong et al., 2003; Bertino and Sandhu, 2005; Zhu et al., 2007; Zhu et al., 2009). The relation between these levels is defined as TopSecret > Secret > Confidential > Unclassified. Doung and Zhang (2008; 2010) define other ordered label sets in their access control system as Protected > Private > Public. On the other hand, unordered labels can be defined as names that are used in the domain of the databases such as Technique, Human Resources, Financial, etc (Sandhu and Samarati, 1994; Bertino and Sandhu, 2005; Zhu et al., 2007; Zhu et al., 2009).

Access operations in Mandatory Access Control depend on two principles: read-down and write-up. Read-down means that users can read only objects at their level or lower. Write-up means that users can write only at that level or higher. However, the write-up principle causes problems for data integrity because a subject in a lower level can write data at a higher level without having the privilege to read data at that level, and so may overwrite data. Some systems overcome this issue by eliminating the write-up concept and making the write process permitted for the same level only (Sandhu and Samarati, 1994; Jeong et al., 2003; Bertino and Sandhu, 2005). This type of access control is not adopted widely by database management systems (DBMSs) because defining and classifying security levels in organisations is a difficult process (Rashid et al., 2010). This type of access control is used in high security systems such as military applications (Li et al., 2005; Zhang and Xue, 2005).

Mandatory Access Control (MAC) were applied to XML databases context by several authors (Cho et al., 2002; Li et al., 2005; Zhang and Xue, 2005; Zhu et al., 2007; Zhu et al., 2009). MAC can be implemented based on XML files, a DTD, or an XML Schema (Cho et al., 2002; Zhang and Xue, 2005; Zhu et al., 2009). The core aim in MAC is to assign labels to both subjects and objects. Objects are elements and attributes in XML database. The labels for subjects are stored separately in a special file. Labels for XML databases objects are also stored in a separate file (Cho et al., 2002; Li et al., 2005; Zhang and Xue, 2005). Zhu et al. (2007; 2009) stored objects labels as attributes of elements in the XML file itself or the XML schema. It appears that this method may consume more storage and time for access. MAC can be used in conjunction with other techniques described later in Section 3.5 (Li et al., 2005).

3.4.3 Role Based Access Control (RBAC)

Role Based Access Control (RBAC) divides subjects (users) according to their roles and responsibilities in the system. Privileges and rights to access objects are assigned to roles and then subjects are assigned to roles depending on

their job (Sandhu and Samarati, 1994; Hitchens and Varadharajan, 2001; Jeong et al., 2003; Zhang et al., 2004; Bertino and Sandhu, 2005; Park and Giordano, 2006; Rashid et al., 2010; Zhao et al., 2010). The management of the authorisation process becomes easier and more effective by being broken down into two steps rather than assigning privileges to subjects directly. If the user's role changes, it will be necessary only to revoke the old role and assign a new role without changing privileges. In addition, the relationship between a role and subject is a many to many relationship that means the role can be assigned to a subject or group of subjects and the subject can have one role or a group of roles (Hitchens and Varadharajan, 2001; Bertino and Sandhu, 2005). Usually Role Based Access Control has a role hierarchy that relates to all the roles in the systems (Hitchens and Varadharajan, 2001; Wang and Osborn, 2004; Zhang et al., 2004; Bertino and Sandhu, 2005; Rashid et al., 2010) .

The administrator defines the role for each user and he can control the authorisation of roles by adding constraints on execution processes. This category of access control is the most popular one in large-scale systems and it has a great influence in the access control area (Zhang et al., 2004; Bertino and Sandhu, 2005; Park and Giordano, 2006; Xing et al., 2010; Zhao et al., 2010; Thimma et al., 2013). However, it does face privilege abuse from internal subjects who reach data sources and misuse their roles (Sandhu and Samarati, 1994; Hitchens and Varadharajan, 2001; Jeong et al., 2003; Wang and Osborn, 2004; Bertino and Sandhu, 2005; Byun et al., 2005; Park and Giordano, 2006; Rashid et al., 2010; Xing et al., 2010). This type as other traditional types cannot easily handle privacy protection (Xing et al., 2010; Sun and Wang, 2011).

RBAC is widely used and simple. It is implemented within XML databases via different models (Hitchens and Varadharajan, 2001; Wang and Osborn, 2004; Zhang et al., 2004; Thimma et al., 2013). It can be applied for both local and server systems since it supports remote access (Hitchens and Varadharajan, 2001; Zhang et al., 2004). These models assign authorisations via

policies for the whole XML file or elements themself. The access processes are based on XPath expressions (Hitchens and Varadharajan, 2001; Wang and Osborn, 2004; Zhang et al., 2004). Zhang et al. (2004) and Hitchens and Varadharajan (2001) depend on using XML Schema and DTD in their models.

3.4.4 Alternative Types of Access Control

The traditional types of access control were described in the previous Sections 3.4.1 – 3.4.3. This Section discusses the other approaches to access control including Attribute Based Access Control (ABAC), Function Based Access Control, and Purpose Based Access Control. Since the Trust Based Access Control approach is the main topic of this thesis and is not yet applied to XML databases, it will be described separately in detail in Chapter 5.

Attribute Based Access Control (ABAC) is a relatively new type of access control (Yuan and Tong, 2005). It depends on using attributes to define the access policy. These attributes include subject attributes, objects attributes, and environment attributes (Yuan and Tong, 2005; Shen, 2010; Zhang et al., 2013). It is designed to cover the features of DAC, MAC, and RBAC (Yuan and Tong, 2005; Shen, 2010; Jin et al., 2012). The identification, security levels, classifications, and roles can be considered as attributes (Yuan and Tong, 2005; Jin et al., 2012). This type can also include other attributes such as time (Yuan and Tong, 2005). As an example of a subject attribute, the user can access the specific data if his age is over 18. The access claim is checked to see whether it meets the defined attributes or not. In the context of XML, XACML (see Section 3.2) efficiently supports the implementation of Attribute Based Access Control (ABAC) for applications such as web services (Yuan and Tong, 2005; Shen, 2010; Jin et al., 2012). Although this type is flexible, it makes the access control policy more complex (Jin et al., 2012). For this reason, it has not been adopted by large scale systems (Shen, 2010).

Function Based Access Control was proposed by Qi et al. (2005) using rule functions. These rule functions are executable code that includes the access control policy. They can be related to the subject, the object, or be general. Many XML documents can be shared using these rules functions. Unlike most approaches it handles both positive and negative authorisations (Qi et al., 2005; Di Vimercati et al., 2008). The object rule functions are similar to the ACL approach (see Section 3.3) and the subject access control functions are similar to the capability approach (see Section 3.3). It appears that this approach may not fit with a system with a complex structure due to its similarity to ACL and Capability approaches. Although the author implemented the approach practically, further investigations are needed to check the efficiency and performance in different situations such as for update privileges.

Purpose Based Access Control restricts the access policy depending on the purpose notation. The purpose is the reasons behind accessing and collecting data. The purposes are organised in hierarchical relations using a tree structure (Byun et al., 2005; Sun et al., 2010; Sun and Wang, 2011). The intended purpose defines a set of purposes and associates them with objects. The access process is based on checking the access purpose and the intended purpose; if they match access is permitted, or otherwise denied (Byun et al., 2005; Sun et al., 2010; Sun and Wang, 2011). Byun et al. (2005) refer to using RBAC and assigning the purpose to the role. The authors state that combining these two approaches makes the access policy more complex. Purpose Based Access Control is applied theoretically to XML databases by Sun and Wang (2010; 2011). They comment that this approach is a first step and still requires further improvements. Lack of implementation means it is impossible to evaluate this type of access control.

Some general points are concluded from describing all these access control types. All of them are aimed at protecting data. With respect to all developed models, the access control suffered from limitations and needs to be improved further and, in some cases, implemented to understand how they work

in practice. As described above, most of the approaches have limited implementation in XML databases. The most important point is that all of these approaches are static, which means defining the access policy and assigning privileges forever. In addition, all of these approaches lack the possibility of capturing misuse and insider threats. They also do not consider the user interaction in the access policies. These reasons lead to the implementation of dynamic and responsive access control for XML databases. Thus, this thesis focuses on applying dynamic Trust Based Access Control for XML databases. The research motivations and contributions are explained in Chapter 5 and the literature review of the Trust Based Access Control is covered in Chapter 4. The next Section describes different techniques used to apply access control for XML databases.

3.5 Access Control Techniques For XML Databases

The majority of traditional and proposed access control approaches for XML databases such as role based, mandatory based, purpose based, and function based focus on processing access to XML files and elements using XPath (Hitchens and Varadharajan, 2001; Wang and Osborn, 2004; Qi et al., 2005; Zhu et al., 2007; Li and Hong, 2008; Landberg et al., 2010; Sun et al., 2010). XPath was discussed in Section 2.6 is used as a basic technique to access a node or set of nodes in XML databases.

An additional technique used to control access to XML databases is the use of views. A view presents data partially based on personalisation and specifications requirements (Abiteboul, 1999; Chen et al., 2002; Kozankiewicz et al., 2003; Balmin et al., 2004; Arion et al., 2007; Roantree et al., 2007; Gire and Idabal, 2008; Yuanbo et al., 2009). It can present data from different perspectives to different users (Abiteboul, 1999; Chen et al., 2002; Yuanbo et al., 2009). The use of views for these purposes originated in relational databases (Elmasri and Navathe, 2003; Teorey et al., 2006; Molina et al., 2009; Elmasri

and Navathe, 2011). Security views are created based on users (Byun and Park, 2010). They depend on users' rights so, for instance, in Mandatory Access Control they are generated according to users' labelling levels (Di Vimercati et al., 2005; Li et al., 2005). The DTD or the XML schema that can be used to create materialised or virtual views (Kozankiewicz et al., 2003; Balmin et al., 2004; Yuanbo et al., 2009). They need high maintenance and much storage (Byun and Park, 2010). The topic of views is large and it is related to other XML databases techniques like optimising queries and updating data. It is mentioned here because it can be used to manage access control but it is beyond the scope of this thesis. This work is designed to manage the simple access to a specific node rather than make the access process more complex with dealing with groups of nodes in views.

Many different models for XML access control to data have been proposed by using two main techniques: node filtering (Yu et al., 2002; Gabillon, 2004; Yu et al., 2004; Gabillon, 2005) and query rewriting (Mohan et al., 2005; Mohan et al., 2006a; Mohan et al., 2006b; Damiani et al., 2007; Damiani et al., 2008; Byun and Park, 2010; Thimma et al., 2013). Both techniques can use the security views. Likewise, other techniques have been used to develop access control systems such as labelling and path indexing (An and Park, 2007; Duong and Zhang, 2008; Duong, 2010; An and Park, 2011). These techniques consider access control from a processing query perspective and do not relate to applying different types of access control to XML databases. These techniques are discussed in the following Sections.

3.5.1 Node Filtering

In simple terms, node filtering means scanning and parsing the whole tree and giving each node a positive or negative sign. This sign is used to indicate whether the access is permitted or denied. The node filtering technique depends on access policies to create a user view that can then be integrated with

user queries (Gabillon, 2004; Gabillon, 2005; Damiani et al., 2008; Byun and Park, 2010).

In the early stages of developing access control for XML databases, node filtering was usually used. Although it is simple, it suffers from limitations. It consumes a large amount of storage space and needs high maintenance since the filtering process is repeated many times for each user or group of users (Damiani et al., 2008; Byun and Park, 2010).

Some traditional access control systems for XML are based on this technique and were developed by Damiani et al. (2000a; 2000b; 2001; 2002), Yu et al. (2002; 2004) and Gabillon (2004; 2005). Researchers who used node filtering extended their work to develop query rewriting; this is described in Section 3.5.2. Some node filtering models can support only the read privilege (Damiani et al., 2000a; Damiani et al., 2000b; Damiani et al., 2001; Yu et al., 2002; Yu et al., 2004) and other models applied both read and write privileges (Damiani et al., 2002; Gabillon, 2004; Gabillon, 2005; Di Vimercati et al., 2008).

A novel privilege, called position privilege, was introduced by Gabillon (2004, 2005) to solve some issues in node filtering techniques. The position privilege makes a separation between the node's existence and its content. In other words, it allows the user to know about the node's existence without knowing the node's label and value. The node has a RESTRICTED label and Figure 3.1 explains the position privilege work (Gabillon, 2004; Gabillon, 2005).

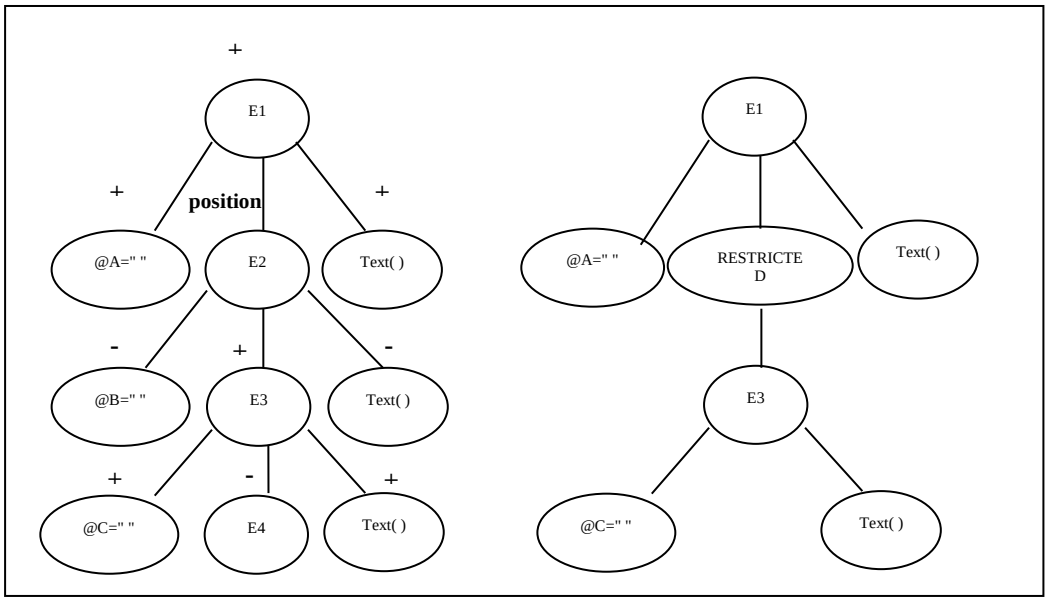


Figure 3.1 Using the position privilege in user view (Gabillon, 2004)

A Compressed Accessibility Map (CAM) approach for XML database access control was proposed to solve the storage problem and speed up the process in this technique (Yu et al., 2002; Yu et al., 2004). This map gathers together nodes that have similar accessible attributes. The compressed tree looks smaller than the original one and has fewer nodes. The model can provide fast determination of the user's rights. The main function of this map is to identify whether access is allowed or not for each user. It can be defined as M=H*U*A, where M refers to the map, H refers to the database tree, U refers to the users, and A refers to the access modes. This approach contains algorithms to find the best compressed accessibility map to reduce the storage space consumed. A compressed accessibility map is created for each user and each access mode (Yu et al., 2002; Yu et al., 2004). Although this approach was designed to reduce the storage space by reducing the tree size, it still consumes a large amount of storage due to the presence of many users' maps (Duong and Zhang, 2008; Duong, 2010).

The authorisation model in the node filtering technique can deal with a single XML document or a group of XML documents by working with a DTD or XML schema (Damiani et al., 2000a; Damiani et al., 2000b; Damiani et al., 2001; Di Vimercati et al., 2005). The access control system can add the access rules to the XML file or XML schema in separate files (Di Vimercati et al., 2005).

Node filtering is one of the two main approaches that describe how the access process is done in XML databases. The other approach, known as 'query rewriting', is described in the next Section.

3.5.2 Query Rewriting

The query rewriting technique depends on transforming possibly unsafe queries into safe ones that can then access the data. This technique slows access (Duong and Zhang, 2008; Duong, 2010; An and Park, 2011). Several models have been proposed based on this technique.

Some systems that use the query rewriting create annotated schemas by using a variety of attributes (Mohan et al., 2005; Mohan et al., 2006a; Mohan et al., 2006b; Damiani et al., 2007; Mohan et al., 2007; Damiani et al., 2008). These attributes include access, condition, dirty. The access attribute provides the 'allow' or 'deny' right to the subject to access the object. The condition attribute includes a number of predicates. The third attribute, 'dirty', indicates that access to some node's descendants may be denied. A user schema view can be easily created from the annotated schema. A finite state automation is used in the rewrite automation process (Damiani et al., 2007; Damiani et al., 2008). Mohan et al. (2005; 2006a; 2006b; 2007) improved this technique by using virtual views based on the security view specification. The system can be applied for both read and write (insert, delete, update) privileges. Figure 3.2 shows all the steps in the query rewriting process.

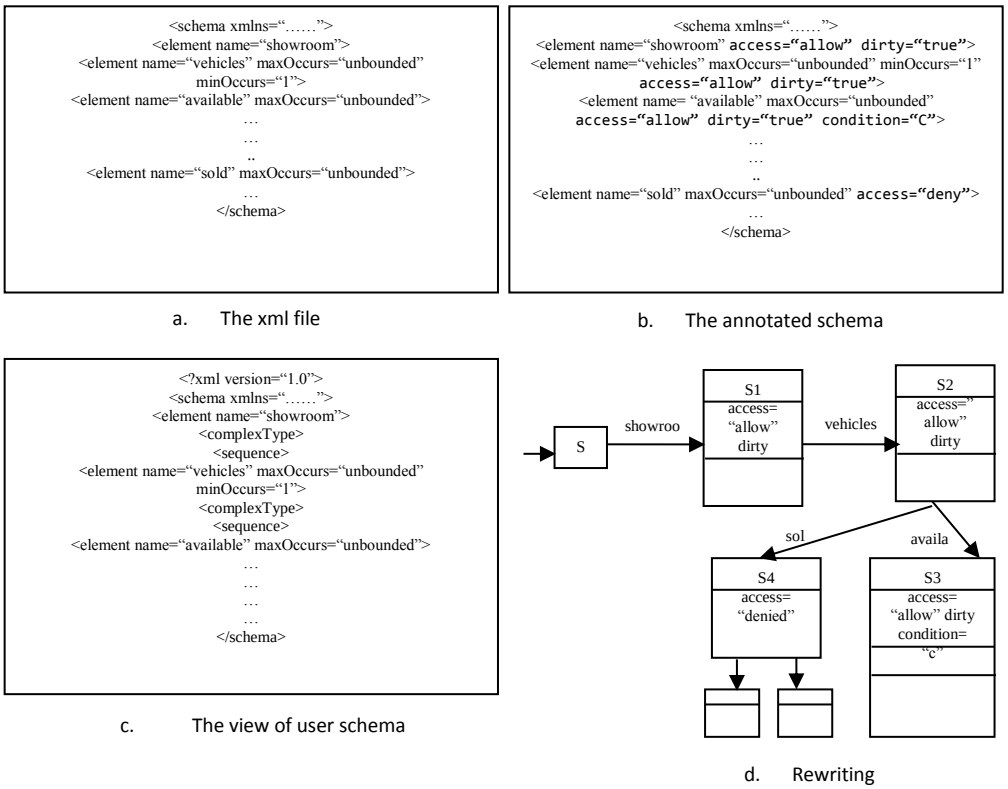


Figure 3.2 The access control model was proposed by Damiani et al. (2008)

3.5.3 Tables and Files Techniques

Some access control models do not depend on either node filtering or query rewriting techniques. These models use a variety of methods that avoid repeated processes for users. They aim to define a standard access control system that is suitable for all users rather than find a particular access mode for each user and so do not repeat the process several times (Kitagawa and Yoshikawa, 2005; Duong and Zhang, 2008; Duong, 2010).

The access control system designed by Kitagawa and Yoshikawa (2005) depends on policy tables. The policy table handles policy rules that are defined based on system strategies. Due to the policy tables' size and their numbers, a simplification and unification process is performed and the access decisions are improved by reducing the time consumed. The policy table consists of two columns: pathID and flag; each one of these tables represents only one role. The

pathID for each node is taken from the path information table, which gives each path a number (ID). The flag in the policy table reflects whether access is allowed (+) or denied (-).

The simplification process is used to minimise the table size by representing only the minimum pathID when sequential numbers have the similar access results. Then the unification stage occurs by creating the role table that consists of role name and roleID. Finally, the unifying policy table (UPT) is generated from the role table and policy tables. The UPT table includes pathID and flag key number. PathID refers to the node in the XML tree and the flag key number is a product of the roleID, which only allows access to this node (Kitagawa and Yoshikawa, 2005). This system is based on many tables, which means using a relational database to implement access control for the XML database. The authors described this technique with read privileges and did not mention write privileges. It appears that implementing the write privilege may cause some difficulties regarding integrity although they did mention this in the paper. For example, the delete processes need to change data in many tables, which may consume much more time.

Duong and Zhang (2008) defined access control concepts in files. The XML Access Authorisation file (XAA) included all XML elements and their access levels. The access authorisation of user groups is defined in the XML Group Authorisation (XGA). In this technique, the access level was classified as: public < private < protected. Many different symbols were used to refer to access levels, such as "#" to refer to the protected level. When this method is compared with the node filtering technique on the basis of the number of nodes scanned and response time, it demonstrates a good performance in terms of speed and accuracy (Duong and Zhang, 2008; Duong, 2010). Using files to define access control rules and concepts is also used by many Mandatory Access Control models (Cho et al., 2002; Li et al., 2005; Zhang and Xue, 2005) as mentioned previously in Section 3.4.2. A file-based technique is used to

implement the Trust Based Access Control in this thesis due to its simplicity and clarity as well as speed.

3.6 Labelling Techniques

In this Section, labelling is described and its types are discussed. The node labelling technique gives each node in the tree a unique label; used to identify and access the nodes. Currently, there are several models for node labelling (Cohen et al., 2002; O'Neil et al., 2004; Duong and Zhang, 2005; Gabillon and Fansi, 2005; Khaing and Thein, 2006; An and Park, 2007; Duong and Zhang, 2008; An and Park, 2011).

The models for labelling schema are either static or dynamic. The static approach is not efficient due to relabelling processes, which consumes much time and requires maintenance. Relabelling process means changing the old labels for all nodes and giving them new ones. The dynamic labelling techniques were designed to overcome these limitations in update processes caused by inserts. Some dynamic labelling models used numbers (Cohen et al., 2002; O'Neil et al., 2004; Gabillon and Fansi, 2005) and others used a mixture of numbers and letters (Duong and Zhang, 2005; Khaing and Thein, 2006; An and Park, 2007; Duong and Zhang, 2008; An and Park, 2011). These models used different forms but the majority of them used the dots "." in their labelling technique (Cohne et al., 2002; O'Neil et al., 2004; Duong and Zhang, 2005; Khaing and Thein, 2006; An and Park, 2007; Duong and Zhang, 2008; An and Park, 2011) although Gabillon and Fansi (2005) used the ordered pair approach (1,1) in their system.

Although these dynamic models try to avoid relabelling; some models still need the relabelling in limited cases (O'Neil et al., 2004). Moreover, some systems suffer from collisions between nodes. A Collision means using the same

label for two different nodes. This causes a problem that affects the system's work and efficiency (Duong, 2010; Khaing and Thein, 2006).

Most existing researchers classify labelling schemes into range based (Li and Moon, 2001; Amagasa et al., 2003) and prefix based (Cohne et al., 2002; O'Neil et al., 2004; Duong and Zhang, 2005; Gabillon and Fansi, 2005; Khaing and Thein, 2006; Duong and Zhang, 2008; An and Park, 2011). However, a few authors discuss other new types based on mathematical approaches. All categories are discussed in the following Sections.

3.6.1 Labelling Scheme Types

3.6.1.1 The Ranged Based Scheme

Range based labelling schemes (interval based labelling- region based labelling) can show the ancestor and descendant relations and the parent and child relations between nodes (Al-Shaikh et al., 2010; Duong, 2010; Xu et al., 2010). This category includes pre/post labelling and containment labelling scheme (Sans and Laurent, 2008; Xu et al., 2010). The pre/post labelling scheme used both pre-order traversal and post-order traversal that are illustrated in Figure 3.3. This labelling scheme generates the node label as pre, post, level (Dietz, 1982; Xu et al., 2010).

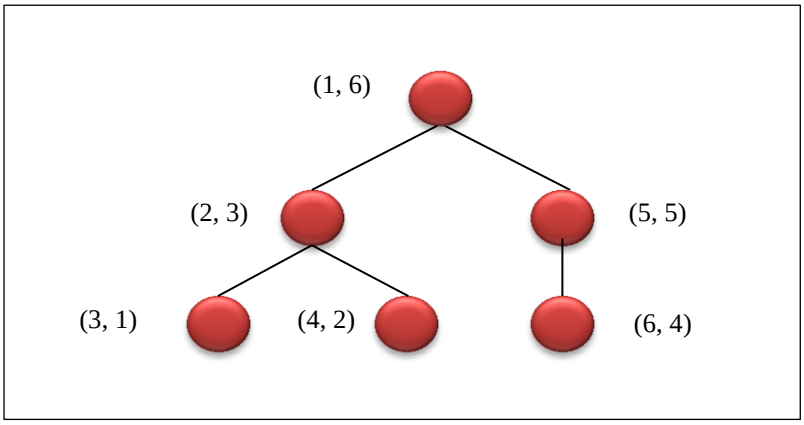


Figure 3.3 Pre-order and Post-order labelling scheme (Dietz, 1982)

The containment labelling technique used the labelling form start, ends, level (Duong, 2010; Xu et al., 2010). Figure 3.4 shows the containment labelling technique. Other ranged based schemes used several labelling forms, which depend on tree traversal such as (order(x), size(x)) (Li and Moon, 2001). The range based scheme type is flexible but it suffers from relabelling due to using sequential numbers. Some solutions were proposed to overcome this problem by pre reserved extra spaces or using floating points (Duong and Zhang, 2005; Al-Shaikh et al., 2010; Duong, 2010).

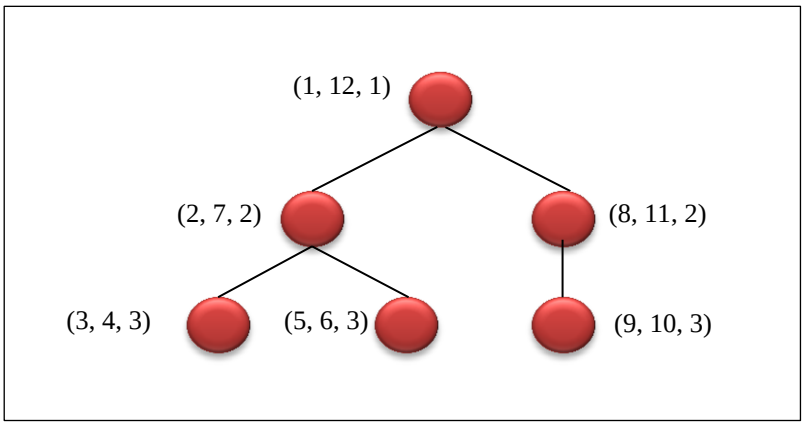


Figure 3.4 The containment labelling scheme (Duong, 2010)

3.6.1.2 The Prefix Based Scheme

The prefix based technique is encoding the tree depth. This technique adds the parent code to the node label as prefix (Sans and Laurent, 2008; Al-Shaikh et al., 2010; Duong, 2010; Xu et al., 2010). The most popular algorithm is the Dewey label (Xu et al., 2009), which is based on a system used by librarians. Figure 3.5 illustrates the Dewey labelling as an example of the prefix based scheme type. The form of this label consists of the parent label and the self-node label. So, ancestor and descendent, parent and child, and the sibling relationship can be derived easily from the prefix based labelling scheme. There are several models that used this kind of labelling scheme, such as LSDX, FLEX, and ORDPATH (Deschler and Rundensteiner, 2003; O'Neil et al., 2004; Duong and Zhang, 2005; Duong and Zhang, 2008). In general, this prefix based

scheme type supports the tree growth. However, the breadth growth leads to increase the size of label (Al-Shaikh et al., 2010). This approach requires relabeling in some cases due to using the prefix point (Duong and Zhang, 2005; Duong, 2010).

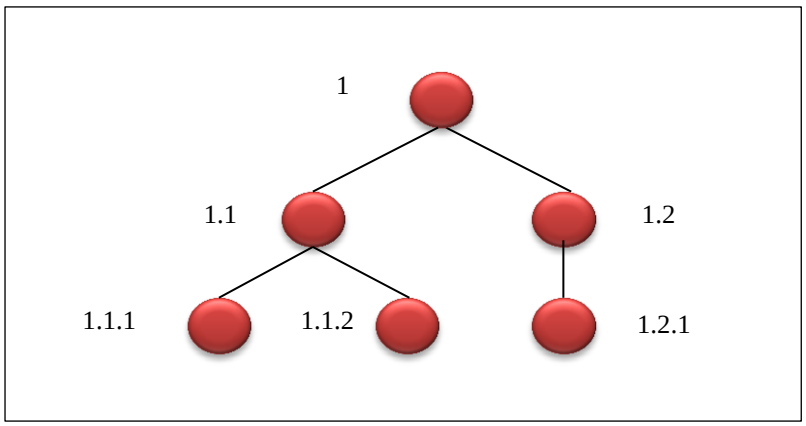


Figure 3.5 Dewey labelling scheme (Xu et al. ,2002)

3.6.1.3 The Mathematical Based Scheme

Although much research focused on developing ranged and prefix based approaches and tackling their problems, some recent research uses novel approaches that are based on mathematical concepts (Kim et al., 2009; Na and Guoqing, 2010; Xu et al., 2010; Zhang and Dong, 2010). Kim et al. (2009) used circular concepts in their labelling approach. They proposed a general labelling form that depends on the concept of angles and applies a rotation angle to the general form when the XML file size is large. This technique restricts the length of the label compared to other techniques that increased the label's length with repeated updates. Alternatively, Xu et al. (2010) proposed a vector code that was represented graphically by using X and Y axes. They then applied and implemented the vector technique to many existing range based labelling and prefix based labelling scheme models and compared the results. In addition, the polar coordinate system that depends on angles and vector concepts was proposed by Zhang and Dong (2010). The label consists of the node's level, the flag for overflow, the start angle, and the end angle. When there is no space to

insert a new node, overflow happens. They solved this problem by following the same technique as LSDX (Duong and Zhang, 2005; Duong, 2010) and adding a letter. Na and Guoqing (2010) classified the XML tree to partitions and gave a mesh partition label to all nodes. They then combined the mesh, prefix, and interval labels. The mathematical approaches are relatively new and need more investigation.

The labelling process can improve the speed to access the node. An and Park (2007, 2011) relate the labelling node model to the access control model directly and other approaches relate them indirectly (Doung and Zhang, 2005; Gabillon and Fansi, 2005; Khaing and Thein, 2006; Duong, 2010).

A direct relationship means the labelling process is used to develop the access control model. An and Park (2007, 2011) designed an efficient control system that depends on node labelling. They defined the prime number product that reflects the users' access to the node. For example in the hospital database, the number 6 is a product of 2 and 3; 2 refer to the patients' group and 3 refers to the doctors' group. They then used this prime number to create the node's label, which is structured as lL1.L2.L3: where l is the number of the level, L1 is the parent node label, L2 is the current node, and L3 is the role based prime number. Some other access control systems use the labelling to improve the query process (Doung and Zhang, 2005; Gabillon and Fansi, 2005; Khaing and Thein, 2006; Doung and Zhang, 2008; Duong, 2010). Doung and Zhang (2008; 2010) integrated between their access model and their labelling model to provide a secure model to query XML databases.

Even though issues with the labelling process have been approached in many different ways, it still suffers from problems that require further investigation. In particular problems of the space required, the speed of queries and relabelling. Labelling topic is beyond the research scope of this thesis but it is mentioned here briefly because it is related to access control.

3.7 Conclusion

This Chapter reviewed work on security in XML databases and related topics. It described the access control types and highlighted both their advantages and disadvantages. Several techniques for accessing XML databases were described. The next Chapter discusses the related work on Trust Based Access Control and explains its concepts.

4 RELATED WORK ON TRUST BASED ACCESS CONTROL (TBAC)

4.1 Introduction

Trust Based Access Control has become an established in many areas, such as networks and virtual organisations (Abdul-Rahman and Hailes, 1997; Cahill et al., 2003; Almenarez et al., 2004; Almenarez et al., 2005; Tran et al., 2005; Almenarez et al., 2006; Lin et al., 2006; Feng et al., 2008; Jia et al., 2009; Ma et al., 2010; Xing et al., 2010; Zhang and Rao, 2010; Zhao et al., 2010; Singh, 2011). Much research has focused on developing and improving it. Trust Based Access Control is discussed in detail here because it is central to the thesis.

This Chapter describes the main concepts of trust. Section 4.2 defines this term and Section 4.3 highlights the trust features. The Trust Value (TV) is described in Section 4.4. The trust relationships between entities can be classified in to direct or indirect trust. Both kinds are discussed in Section 4.5. The calculation of Trust Value (TV) in several applications is discussed in Section 4.6. Finally, conclusion summarises the main points.

4.2 Trust

The term ‘trust’ is defined in many fields such as sociology, psychology, mathematics, and economics. A related definition to the research context is defined by Azzedin and Maheswaran (2002) as follows:

"Trust is the firm belief in the competence of an entity to act as expected such that this firm belief is not a fixed value associated with the entity but rather it is subject to the entity's behaviour and applies only within a specific context at a given time" (Azzedin and Maheswaran, 2002).

Trust depends on beliefs, operations, and recommendations (Almenarez et al., 2004; Tran et al., 2005; Feng and Yang, 2010; Singh, 2011). It requires effort and time to achieve but it can be lost quickly and easily (Almenarez et al., 2006; Lin et al., 2006). Trust is taken from the real world and applied to the digital world. Trust, as used in the access control, means that subjects can trust entities such as other subjects, applications, and firms on the basis of past history, operations, behaviour, experience, and recommendations over time (Abdul-Rahman and Hailes, 1997; Cahill et al., 2003; Almenarez et al., 2004; Almenarez et al., 2005; Almenarez et al., 2006; Lin et al., 2006; Feng et al., 2008; Jia et al., 2009; Ma et al., 2010; Zhao et al., 2010; Singh, 2011).

4.3 Properties of Trust

There are many advantages of using trust. The main feature is that it is dynamic; Trust Values are changeable and can be increased or decreased according to the subjects' behaviour, history, and operations. It is subjective, which means a subject can be trusted to different degrees by other subjects. Each subject trusts itself by default but trust relationships are asymmetrical, which means the relationship can be different in different directions. For example, subject A can trust subject B but at the same time subject B does not trust subject A. It is transitive under some conditions, which means it can be transferred from one entity to others. For example, subject A trusts subject B and at the same time subject B trusts subject C. If subject A trusts subject B as leader, then subject A can trust subject C indirectly. In addition, it is dependent on both past and present time and on context (Abdul-Rahman and Hailes, 1997; Cahill et al.,

2003; Almenarez et al., 2004; Ryutov et al., 2005; Lin et al., 2006; Xing et al., 2010; Zhao et al., 2010; Singh, 2011).

4.4 Trust Value (TV)

Trust Value in the digital world can be measured in several ways. It can be in a range between 0 to 1 where 0 means distrusted and 1 is trusted (Almenarez et al., 2004; Almenarez et al., 2005; Tran et al., 2005; Almenarez et al., 2006; Feng et al., 2008; Jia et al., 2009; Zhang and Rao, 2010; Singh, 2011). It also can be defined using real numbers such as between 1 to 10 where 1 means low trust and 10 is high trust (Abdul-Rahman and Hailes, 1997; Zhao et al., 2010). It can be described by using levels such as L1, L2... L5 where L1 means distrusted and L5 means completely trusted (Lin et al., 2006; Singh, 2011). Trust Value can be called trust degree when using a specific range value, and trust level when using named levels. The concept of a trust degree is more frequently adopted in practice (Abdul-Rahman and Hailes, 1997; Almenarez et al., 2004; Almenarez et al., 2005; Tran et al., 2005; Almenarez et al., 2006; Feng et al., 2008; Jia et al., 2009; Ma et al., 2010; Xing et al., 2010; Zhang and Rao, 2010; Zhao et al., 2010) than levels. Levels are used by limited systems that know the number of required levels and use a specific defended range of levels (Lin et al., 2006; Singh, 2011). The majority of models were developed based on trust degree because it allows a wide range of values and is more flexible. As when using real numbers, there is no difficulty in adding more values.

Trust Based Access Control depends on a trust management system, which automatically calculates and updates the Trust Values of users. Trust Values rely on users' behaviour, users' history, users' credit, and users' operations. Users can access resources through Trust Values and levels (Ryutov et al., 2005; Almenarez et al., 2006; Lin et al., 2006; Ma et al., 2008; Xing et al., 2010).

4.5 Trust Relationships

Trust relationship between two entities can be categorised as either direct or indirect (Abdul-Rahman and Hailes, 1997; Almenarez et al., 2004; Tran et al., 2005; Singh, 2011). Direct trust relations depend on the past interactions, experiences, and operations between two subjects without any other external resources. Since not all subjects know each other, each direct relation can be assigned an initial value and then change over time depending on actions. The direct trust relation is reliable when evaluating the Trust Value. On the other hand, the indirect relation depends on recommendations from a third party. For example, subject A is a recommender who recommends subject B to interact with subject C because there was a good experience between subject A and subject C in the past (Abdul-Rahman and Hailes, 1997; Almenarez et al., 2004; Almenarez et al., 2005; Tran et al., 2005; Ma et al., 2008; Singh, 2011). Naturally, direct trust is more reliable than indirect trust to evaluate the user behaviour (Tran et al., 2005; Ma et al., 2008). The next Section describes how the Trust Value is calculated in different models.

4.6 The Calculation of Trust Value (TV)

Trust Based Access Control models use different techniques to calculate Trust Values depending on their system design, needs, and goals. Each system finds the most effective factors in Trust Value and then includes them in the calculations. Some models for Trust Based Access Control depend on both direct and indirect relations to calculate the Trust Value (Almenarez et al., 2004; Almenarez et al., 2005; Tran et al., 2005; Almenarez et al., 2006; Zhang and Rao, 2010; Singh, 2011). Other models are based only on direct relations and interactions between entities in calculation processes (Xing et al., 2010; Zhao et al., 2010). Other factors such as reputation, contribution, and domain trust are also included in calculation evaluations (Abdul-Rahman and Hailes, 1997; Tran et al., 2005; Feng et al., 2008; Ma et al., 2008; Singh, 2011).

In the context of networks, many methods have been proposed and used to calculate the Trust Value between entities. These methods and equations are discussed in this Section. Almenarez et al. (2006) used a mathematical model to calculate and recalculate the Trust Value depending on the past and present operations. This model is based on equation (1).

$$T_i = \begin{cases} T_{i-1} + \omega . V_{a_i} (1 - T_{i-1}) & V_{a_i} > 0 \\ T_{i-1} \left(1 - \omega + \omega . V_{a_i}\right) & else \end{cases} \qquad \textbf{(1)}$$

In formula (1), T_i means the recalculated Trust Value and T_{i-1} means the previous value. Where ω is a difference percentage that connects with the subjects' disposition in the past and the present. V_{a_i} is the weight variable that depends on past operations and is calculated according to action weight. They developed a probabilistic model to calculate the action value based on evaluating user behaviour. This model uses Bayes' theorem. It was recommended by authors to handle risk management in their future work (Almenarez et al., 2006).

The trust management system designed by Zhang and Rao (2010), which includes four parts: subject manager, trust manager, action monitor, and recommendation manager. The subject manager handles subject information such as subject trust value. The trust manager calculates the Trust Value using equation (1), which is similar to the previous model. The action monitor records subjects' behaviour and the recommendation manager to handle the indirect trust relationship.

Lin et al. (2006) discussed the formal equation to calculate the Trust Value between two entities.

$$\boldsymbol{T_{p1,p2}} = \frac{\sum_{i=1}^{n} [o_i(B_i) - e_i(B_i)]}{\mathbf{t}} \qquad \textbf{(2)}$$

This equation (2) calculates the Trust Value between two parties P1 and P2 from P1's prospective. Where t is the time for previous operations between them; i reflects the number of interactions; B_i is behaviours interaction; o_i is the

observed value for these behaviours; e_i is expected value for these behaviours (Lin et al., 2006).

As mentioned above, any related factor can be included in the calculation. The contribution factor that is based on measuring the shared data in megabytes is included in calculating Trust Value by Tran et al. (2005). Likewise, Feng et al. (2008) include the reputation evaluation in the Trust Value, as in equation (3). The reputation is based on the evaluation of past interactions for the user by other users.

$$Trust = \alpha t + (1 - \alpha)T\ ,\ 0 < \alpha < 1 \qquad \textbf{(3)}$$

Where α refers to the weight given by the system, *t* means the local reputation and *T* is global reputation. The local reputation reflects the evaluation process between two nodes while the global reputation covers the whole network, which means all nodes evaluate the specific node.

The trust calculation can be calculated simply by adding the direct Trust Value and the indirect Trust Value and multiplying each value with its weight (Singh, 2011).

$$TV = wdt * dt + wrt * rt \qquad \textbf{(4)}$$

Equation (4) shows that Trust Value depends on direct Trust Value dt, recommended Trust Value rt, and their weights wdt and wrt. This simple concept in calculating Trust Values is used in the proposed model in this thesis. It is discussed further in Chapter 7.

The maximum and minimum values are used to keep the Trust Value in the correct range (Xing et al., 2010; Zhao et al., 2010). Some Trust Based Access Control modules use flowcharts to explain how their systems work step by step, from request to access resources (Tran et al., 2005; Feng et al., 2008; Ma et al., 2008). Some models implement their approach by using SUN's

XACML API that depends on XACML (Almenarez et al., 2005; Zhang and Rao, 2010; Singh, 2011). XACML is used to write and describe the access policies in XML. This was discussed earlier in Section 3.2.

Trust Based Access Control can be mixed with other types of access control to add dynamic features to the system such as RBAC (Xing et al., 2010; Zhao et al., 2010) and ABAC (Zhang et al., 2013). Yuan and Tong (2005) mentioned that mixing between access control types to utilise all their advantages is a normal process.

From discussing the existing method to calculate the Trust Value (TV), it appears that there is no golden rule for calculations. However, some ideas in different approaches can be changed slightly and adopted to fit the system design in this thesis.

There is no published literature relating to the use of Trust Based Access Control in the context of databases in general and in the XML area in particular except the research papers that relate to this thesis. The aims of applying trust to XML databases were described in Chapter 5. The system design is described in Chapters 6, 7 and 8.

4.7 Conclusion

This Chapter described the main concepts in Trust Based Access Control. It classified the trust relationship into direct trust and indirect trust. Several methods to calculate Trust Value were discussed. Although, these models described Trust Based Access Control in network area, the general aspect can be adopted to XML databases. The next Chapter described the research motivations and objectives of applying Trust Based Access Control for XML databases.

5 THE RESEARCH HYPOTHESIS

5.1 Introduction

This Chapter presents the motivation for this research in Section 5.2 and the research hypothesis in Section 5.3. In Section 5.4, the main objective, which is to develop a dynamic access model that is responsive to an evaluation of users' history, is explained. The model tracks users' errors and bad transactions over time and updates their privileges dynamically. The system prevents outsiders' attacks as well as insiders' malicious processes, effectively preventing users from taking advantage of their role.

5.2 Research Motivation

In this Section, the various motivations behind this research are highlighted. The discussion starts by explaining in Section 5.2.1 the importance of using XML databases as opposed to traditional databases. Section 5.2.2 then explains why the research scope is concerned with the security issues in XML databases. Section 5.2.3 motivates the real need to improve access control for both outsiders and insiders. Overall, the motivation for this research is to develop and improve security in XML databases.

5.2.1 The Importance of XML Databases

In the last decade XML has become well established and used in a wide range of areas and applications such as the web, businesses, information systems, and databases (Abiteboul et al., 2000; Champion, 2001; Tidwell, 2002; Oqbuji, 2004b; Vakali et al., 2005; Anderson, 2008; Jonge, 2008; Whatley, 2009; W3C, 2010; Palani, 2011; Sun and Wang, 2011; Abd El-Aziz and Kannan,

2012b; Abd El-Aziz and Kannan, 2012c; Noaman and Almansour, 2012; Verma et al., 2012; Desai, 2013; Vela et al., 2013; W3Schools, 2013a). Due to the recent increase in their usage, much research has been undertaken to improve their efficiency. XML is used to store, transfer, and manipulate data. It has many advantages; it is readable for both humans and machines. It is flexible, simple, and self-descriptive (Abiteboul et al., 2000; Champion, 2001; Tidwell, 2002; Vakali et al., 2005; Jonge, 2008; Whatley, 2009; W3C, 2010; Palani, 2011; W3Schools, 2013a). This was discussed further in Chapter 2. As a result of this flexibility, the use of XML databases can be expected to improve and develop.

5.2.2 Security in XML Databases

Having data is a power but it must be dealt with in an appropriate and accurate manner (Griffin et al., 2012). Much of the research on XML focuses on storage strategies and query performance. Although data storage and retrieval techniques are important, so is security and, in comparison, this seems to be a neglected research area. XML databases are multi-user systems, meaning they can be accessed by millions of users, and they can provide a huge amount of data. In all applications and especially in platforms such as business and medical applications, XML databases can contain sensitive, personal, and important data. Confidential data needs to be protected and saved in a secure environment for legal reasons and in order to prevent loss or misuse. Security for XML databases is therefore crucial in protecting data from unauthorised processes and misuse (Sun and Wang, 2011; Griffin et al., 2012; Noaman and Almansour, 2012; Verma et al., 2012; Desai, 2013).

One of the main approaches to guarantee security in any system, not just XML databases, is to apply access control. The access control model manages access to data and prevents unauthorised processes (Murata et al., 2006; Sun and Wang, 2011; Verma et al., 2012; Desai, 2013). There has been extensive research in this area but still there are many points that need to be investigated.

5.2.3 Trust Based Access Control (TBAC)

Many different models for XML database access control have been proposed and developed (see Chapter 3). They can be categorised into three core categories: Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role-Based Access Control (RBAC) (Sandhu et al., 1996; Hitchens and Varadharajan, 2001; Wang and Osborn, 2004; Zhu et al., 2007; Zhu et al., 2009; Xing et al., 2010). There are many other types that are non-traditional, such as function based access control and purpose based access control (Qi et al., 2005; Sun et al., 2010; Sun and Wang, 2011; Fiebig et al., 2012). Some of these models have been applied to provide a secure environment for XML databases. Most traditional access control models protect data from the malicious activities of outside users but cannot protect the data from insiders. They cannot easily provide protection for privacy data (Chagarlamudi et al., 2009; Xing et al., 2010; Sun and Wang, 2011). Research has highlighted that damage caused by insiders, who know the system, is more harmful than that of outsiders (Park and Giordano, 2006; Xing et al., 2010). Moreover, internal users may abuse their role and take advantage of their position in the system. The insider threat is a huge topic in data security and many methods have been proposed to identify misuse behaviour (Yi and Brajendra, 2003; Chinchani et al., 2005; Chagarlamudi et al., 2009); yet there has been no work by other authors on dynamic updates to access privileges in relation to trust for XML databases.

Trust Based Access Control is established and used in many areas, such as networks and virtual organisations. It depends on a trust management system, which automatically calculates and updates the Trust Values of users. Trust Values rely on users' behaviours, histories, credit, and operations. Users can access resources through Trust Values and levels (Cahill et al., 2003; Bhatti et al., 2004; Almenarez et al., 2006; Lin et al., 2006; Feng et al., 2008; Ma et al., 2008; Lang et al., 2009; Xing et al., 2010; Farooqi and North, 2011a; Farooqi and North, 2011b; Singh, 2011; Farooqi and North, 2012c; Farooqi and North,

2012a; Farooqi and North, 2012b; Farooqi and North, 2012d; Farooqi and North, 2013). This was discussed in more detail in Chapter 4.

5.3 The Research Hypothesis

Working in the light of motivations in the previous Section, this research aims to test the following hypothesis:

"Since the Trust Based Access Control (TBAC) approach has been applied successfully in many areas, such as networks and virtual organisations, it may also improve security in the XML database research field by providing security to protect sensitive and confidential data from misuse by both outsiders and insiders while not restricting appropriate access."

In this thesis, this hypothesis is investigated by a practical implementation that is tested and evaluated. The implementation covers many concepts: calculating trust factors, developing log files, and managing access processes to XML databases. Then, it has been tested and evaluated with several XML databases of different sizes and structures as well as with different numbers of users.

5.4 The Research Objectives and Contributions

Considering the problems of the traditional access control model mentioned in Section 5.2.3 and applying the research hypothesis that is formalised in Section 5.3, this research has three objectives; they are described in the following Sections.

5.4.1 Appling the Trust Based Access Approach to XML Databases

Since the trust based approach is considered to be one of the new types of access control systems (Lin et al., 2006; Ma et al., 2008), its benefits have not

yet been applied to XML databases (Farooqi and North, 2011a; Farooqi and North, 2011b; Farooqi and North, 2012c; Farooqi and North, 2012d; Farooqi and North, 2012a; Farooqi and North, 2012b; Farooqi and North, 2013). Applying trust factors may make the XML databases' environment more reliable. This mainly depends on using evaluation processes to prevent misuse and encapsulate the access process to make it more secure. Including a trust notion in access procedure makes this approach quite realistic and simulates the trust concept in the real world.

5.4.2 Extending Dynamic and Automatic Access Control to XML Databases

One of the most important features of Trust Based Access Control is that it is dynamic. Compared with traditional approaches to access control, which are static, Trust Based Access Control can make access systems responsive and active (Abiteboul et al., 2000; Champion, 2001; Vakali et al., 2005; W3C, 2010; Xing et al., 2010; Farooqi and North, 2011a; Farooqi and North, 2011b; Sun and Wang, 2011; Farooqi and North, 2012c; Farooqi and North, 2012a; Farooqi and North, 2012b; Farooqi and North, 2012d; Farooqi and North, 2013; W3Schools, 2013a). This approach aims to use a trust management system that automatically calculates users' Trust Values. The Trust Values are then updated according to an evaluation of the user's history.

5.4.3 Improving the Access Control Security Level for XML Databases and User Performance

This model improves data security by evaluating users' histories and operations. This approach extends the established work by considering errors when calculating Trust Values. Therefore, users' permissions and privileges can change in response to their behaviour (Farooqi and North, 2011a; Farooqi and North, 2011b; Farooqi and North, 2012c; Farooqi and North, 2012a; Farooqi and North, 2012b; Farooqi and North, 2012d; Farooqi and North, 2013). This

approach leads to a side effect that is concerned with users' performance. While simultaneously increasing the level of security, this approach can be expected to improve the quality of users' performances by recording their operations.

5.5 Conclusion

It is important to tackle security problems in XML databases to reduce misuse and attacks. Improving access control work is vital to protect the sensitive data and provide a secure environment for users. This approach aims to combine detecting insider threats and improving access control by using trust-based access control. It proposes using trust notions to protect personal data and provide a range of values for accessibility to data. It aims to evaluate users' histories of errors and bad transactions and change their access depending on their behaviour. It attempts to improve the access control performance for XML databases by providing a dynamic and automatic system.

This Chapter endeavoured to highlight the research motivations and define the thesis' objectives. The research hypothesis is formulated by applying Trust Based Access Control for XML databases. The next Chapter gives an overview of the design Trust Based Access Control for XML databases.

6 TRUST BASED ACCESS CONTROL (TBAC) FOR XML DATABASES

6.1 Introduction

Trust based access is an established technique in many fields, such as networks and distributed systems, but has not previously been used for any sort of databases. As mentioned in Chapter 4, in Trust Based Access Control, user privileges are calculated dynamically depending on the user's Trust Value. Applying the technique to XML databases might be expected to have advantages over current techniques.

This Chapter presents Trust Based Access Control for XML databases. It discusses the system generally before going into detail in Chapters 7 and 8. In this Chapter, a general overview of the system is given in Section 6.2. Then, in Section 6.3, the system's main components are explained and their functions are described. Besides showing the system concepts, the rules to manage exceptional situations are defined in Sections 6.4 and 6.6. The Chapter ends with a general conclusion that leads to the following Chapters – 7 and 8 – which explain the system's components in detail.

6.2 The System Overview

In order to improve security and provide dynamic access control for XML databases, Trust Based Access Control for XML databases has been developed. It aims to provide secure access control by detecting insider threats through evaluating users' operations over time. Trust Based Access Control for XML databases manages the access policy depending on users' trustworthiness

and may prevent unauthorised processes, malicious transactions, and misuse from insiders. Outsiders who are not related to the system can be assumed to have no access right but outsiders impersonating insiders should be detected. Trust scores are updated on the basis of users' histories. Privileges are automatically modified and adjusted over time depending on user behaviour to deal with insider threats.

Trust Based Access Control for XML databases is based on direct trust and ignores indirect trust (see Section 4.5). Direct trust focuses on users' operations and errors. Indirect trust depends on recommendations and therefore is largely irrelevant in this context. In this system, some specific operations are defined for monitoring misused. Errors are even though they do not reflect malicious intent. In real life, the person who makes many mistakes is probably not reliable. Such a person could not be trusted to handle important transactions and access sensitive data (Trochim, 2006; Mobley, 2011). The next Section describes the system's components in outline and explains the main functions of each part.

6.3 The System Components

In this Section, a Trust Based Access Control module for XML databases is described. Users access the system through the simple user interface. The user interface receives the access request to the XML database as an XML query then sends it to the system. The module consists of two main parts: the trust module and the access control module. The trust module is responsible for recording errors and bad transactions, evaluating them, and calculating the new Trust Value. The access control module is responsible for the access permission policy and access decisions. Both modules are explained in detail in the next two Chapters 7 and 8. The system uses the concept of designed structure proposed by Zhang and Rao (2010) as a guide to design its structure. The architecture of Trust Based Access Control for XML databases is shown in Figure 6.1.

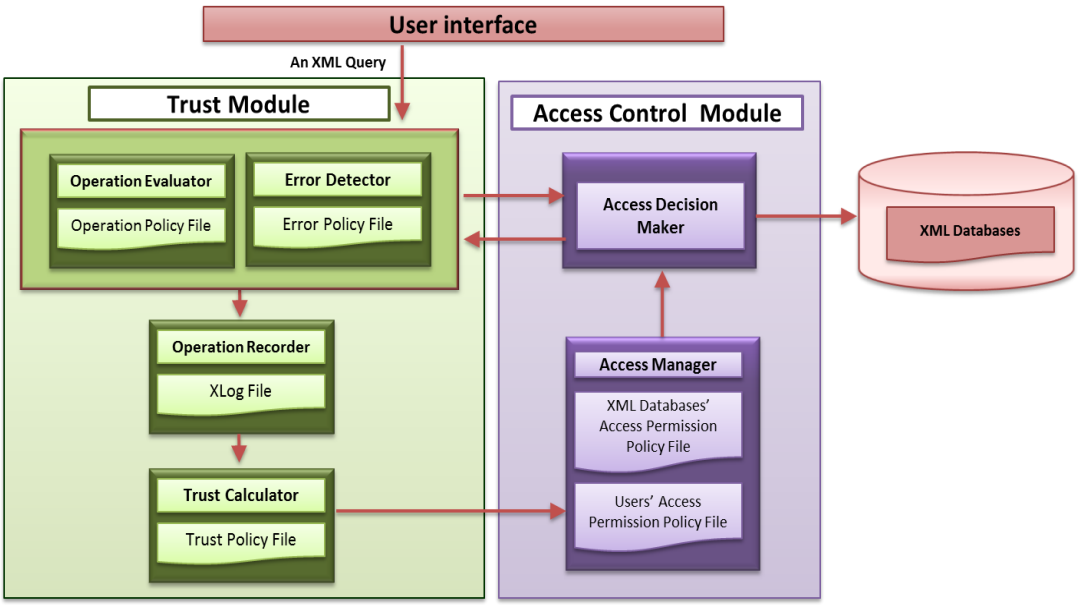


Figure 6.1 The structure of Trust Based Access Control for XML databases

6.3.1 The Trust Module

The trust module is constructed of many parts that work together to achieve the main goal of calculating users' Trust Values. These parts consist of an error detector, an operation evaluator, an operation recorder, and a trust calculator. Both the error detector and operation evaluator work in the light of the error policy file and the operation policy file and capture bad operations and errors. Each of these policy files has the role of defining what an error or a bad transaction is. The operation recorder records errors and bad transactions in the XLog file. The XLog file is designed to be dynamic and to be stored only temporarily for a set period to reduce storage and improve search performance. The trust calculator uses the data recorded in the XLog file in its calculation. The main goal of the trust calculator is to compute a new Trust Value that depends on the user's history of bad transactions and errors.

6.3.2 Access Control Module

The access control module consists of the access manager and the access decision maker. The access manager deals with access permission policies that primarily depend on Trust Value (TV). These policies are divided into subject policy and object policy. A Trust Value is assigned to the subject and the object policy concentrates on giving each item of data the appropriate Trust Value.

The access decision maker handles the XML query and then either permits or denies the request. The final decision depends directly on defined access permission policies in the access manager. The Trust Value (TV) of the user is compared with the Trust Value (TV) of the required XML data. If the Trust Value (TV) for users equals or is larger than the Trust Value (TV) of data, then the user is allowed to access the data; otherwise access is denied. The whole system combining both modules is explained in the next Section.

6.4 The System Processes

The trust module is connected to the access control module to form the complete system. The system processes can be characterised into two main classifications: access supervision and trust maintenance. The access supervision process is always run for each access to the system. The trust maintenance process occurs frequently but depends on the organisation's policy for updating the users' Trust Value, therefore it could run weekly, daily or hourly.

The access supervision process contains a series of small processes. This series starts by receiving an XML query and detecting errors and/or bad transactions, then recording them in the log. The trust maintenance process also consists of a sequence of small processes. These processes evaluate errors and bad transactions, calculating a new Trust Value and updating users' privileges. The whole system is illustrated in Figure 6.2.

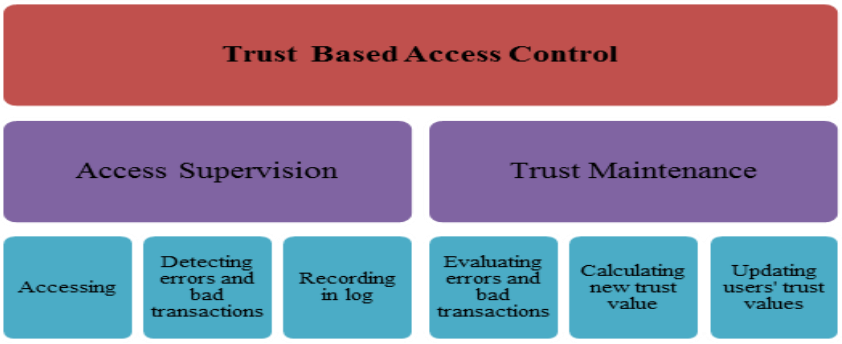


Figure 6.2 The system's processes

6.5 The Objectives of Policy Files

The policy files contain rules. In this Section, the goals of using the policy files are described. The policy rules depend on the organisation policy that differ from one organisation to another. Organisations that contain much sensitive and personal data, such as banks, normally use strict policy rules to provide high level of security to the system. Other organisations may not be concerned about security issues but may focus on processes' speed, which need more flexible and simple policy rules.

In this experimental Trust Based Access Control for an XML databases system, some very basic policy rules are recorded in policy files but they can easily be extended to cover many other complicated policies. The system divides the policy rules into sub rules and records them in individual policy files. The idea of this division is to make the policy rules clear and easy to update and change. Each policy file contains a group of rules related to the specific part in the system.

The policy file is a standard XML file and has defined tags that related to the recorded policy rules. Policy files are written in XML even though there are other access policy languages, which are mentioned in Chapter 3, because these languages do not fit well with Trust Based Access Control properties. These languages depend on DAC, RBAC and ABAC. Furthermore, XML is the

original language and other access languages are derived from it. Using XML makes the policy file clear, easy to use, and flexible; simultaneously, ensuring the file type is consistent with the rest of the system.

There are five different policy files: error, operation, trust, XML database's access permission, and user access permission policy files. The error policy file and the operation policy file have the rules to define what an error and a bad transaction is. They are used by the error detector and the operation evaluator to capture errors and bad transactions. The trust policy file provides the rules to calculate the Trust Value used by the trust calculator. Both the XML database access permission policy file and user access permission policy file are required by the access manager. The XML databases' access permission policy file contains the trust rules to access XML databases by assigning each node an appropriate Trust Value. The user permission access policy file contains the Trust Value for each user in the system. This file is relatively dynamic because the Trust Values for users change over time according to their behaviour. The structure and the content of each policy file are described with the related part of the system in Chapters 7 and 8.

6.6 Boundary Management

Boundary management aims to define basic rules to avoid anomalous situations that may occur in the system over time. As any database's system, the administrator manages the policy rules mentioned in the Section 6.4 and assigns the Trust Values for data and assigns users initial Trust Values based on their roles. The administrator must be authorised to handle any exceptional circumstances that may have occurred in the system.

One of the critical points in the system is the handling calculation Trust Value for users. Since, the Trust Value is dynamic and updated over time; it may cause other access problems such as a Trust Value dropping until it blocks the user's access completely. The boundaries for Trust Values are designed to

control the change in the Trust Value and to make sure the user's access is within the right ranges. There are two main boundaries: the maximum and the minimum. Both values depend on organisational policy and are related to the role of users in the system. The maximum value is the highest Trust Value that the user can reach and the minimum value is the lowest Trust Value the user can fall to.

These boundaries control the powers and permissions of the access in the system. The maximum ensures that user access only the authorised data and the minimum prevents blocking data. This point is adopted from models proposed by Xing et al. (2010) and Zhao et al. (2010). For example, the Trust Value for the manager could be between the maximum value 1 and the minimum value 0.75 (1>= TV>=0.75). Within this range the Trust Value for the manager can change according to behaviour. Any member of staff whose Trust Value approached the minimum too often could be considered a threat. Boundary management can be easily modified to cover other organisations' strategies to handle future risks that the system may face.

6.7 Conclusion

This Chapter described the general approach to Trust Based Access Control for XML databases. It showed the system's components and explained their main functions. The main rules, which are easily modified, for the system policy and risk management were defined. The system outlined here consists of two main modules: the trust module that is described in detail in Chapter 7 and the access control module that is fully described in Chapter 8.

7 THE TRUST MODULE

7.1 Introduction

As mentioned in the previous Chapter, Trust Based Access Control for XML databases consists of two main modules: the trust module and the access control module. In this Chapter, the trust module and its components are described in detail. The outlines of the module are given in Section 7.2. Then each component of the module is explained separately. Section 7.3 shows how the operation evaluator works using the operation policy file. The error detector and the rules of defining errors are described in Section 7.4. In Section 7.5, the operation recorder is mentioned, the recording process in the XLog file is described, and the structure and features of the XLog file are highlighted. Then, the calculation of Trust Value is described in Section 7.6. Finally, a conclusion, summarising the main points, is given in Section 7.7.

7.2 The Trust Module Overall

The trust module is the main part of the Trust Based Access Control system for XML databases. It receives XML queries from users through the user interface, evaluates their queries and calculates their Trust Value. The evaluation process depends on the users existing Trust Value and new bad transactions and errors. After calculating the new Trust Value for the user, it will send the Trust Value to the access control module to update the user's privileges. The trust module aims to:

- Detect bad transactions and errors in the XML query using policy rules for errors and bad transactions.
- Evaluate bad transactions and errors and assign appropriate weights to them.

- Calculate the user's new Trust Value using the policy rules.

This module consists of many parts: the operation evaluator, the error detector, the operation recorder, and the trust calculator. Each part has its functions and works in the light of the related policy rules. All these parts are connected together to achieve the main goal of calculating Trust Values for users. These components are described in detail in the next Sections.

7.3 The Operation Evaluator

The operation evaluator is a component of the trust module. It handles XML queries, checks processes, and captures bad and malicious transactions. The evaluation process for the XML query depends on the policy rules defined in the operation policy file. It is integrated with the access decision maker in the access control module in that they work coherently together. It shares the Trust Value for users and data with the access decision maker allowing it to detect the unauthorised bad transactions. The operation policy file contains the rules which define a bad transaction. In the experimental Trust Based Access Control for XML databases, this file contains only five basic types of bad transactions. These rules are:

- Read unauthorised node: this rule means that the user tries to access the content of a node for which he does not have the right to.
- Write unauthorised node: this rule aims to detect transactions when the user tries to write a new node or update the content of an existing node for which his Trust Value is insufficient.
- Delete unauthorised node: this rule captures the situation when the user attempts to delete a specific node but is not allowed to by his Trust Value.
- Delete root node: this rule discovers if the user intended to delete the root node in the XML database, which would cause catastrophic damage to the XML database.

- Delete parent node with existing children: this rule means that the user wanted to delete a node that has children. Such a transaction affects the XML database's structure and content and so is not permitted.

These operation policy rules aim to detect unauthorised transactions (read- write-delete) and transactions that cause damage to the structure and content of the XML database. They can be easily extended to cover other situations to detect myriad bad transactions. Organisations can define these rules according to their system strategies.

As mentioned in Chapter 6, the operation policy file is written in XML. It is treated as a standard XML file. Some specific defined tags are used to record rules. The root node is defined using <Bad Transactions>. Then each bad transaction rule is defined using <Transaction>. This transaction tag contains two sub elements: <ID> and <Type>. The <ID> is the identifier for each rule. The <Type> defines the rule of each bad transaction. Figure 7.1 shows the structure of the operation policy file.

```
<Bad Transactions>
<Transaction >
<ID> 1 </ID>
<Type> Read unauthorised node </Type >
</Transaction>
<Transaction >
<ID> 2 </ID>
<Type> Write unauthorised node </Type >
</Transaction>
<Transaction >
<ID> 3 </ID>
<Type> Delete unauthorised node </Type >
</Transaction>
<Transaction >
<ID> 4 </ID>
<Type> Delete root node </Type >
</Transaction>
<Transaction >
<ID> 5 </ID>
<Type> Delete parent node with existing children</Type >
</Transaction>
</Bad Transactions>
```

Figure 7.1 The operation policy file

7.4 The Error Detector

The error detector is part of the trust module. The error detector aims to capture errors in users' transactions. The process of detecting errors depends on the policy rules defined in the error policy file. In general, it works similarly to the operation evaluator. It receives the user queries, checks their accuracy, and detects errors in transactions. Like the operation evaluator, the error detector works with the access control module through sharing the Trust Value for users and data with the access decision maker to detect errors in transactions. When an error is detected, it is included in the evaluation process for user behaviour because it reflects on the user reliability when accessing important and sensitive data.

The error policy file includes the policy rules of defining errors. Only three simple types of errors are used in the practical Trust Based Access Control for XML databases:

- Read non-existent node: this rule means that the user wanted to access a node that is not in the XML databases.
- Write non-existent node: this rule discovers if the user aimed to add or update a node that is not defined in the structure of the XML database. A normal user cannot give a Trust Value to the non-existent node because that is the administrator's responsibility. The user can add a new node that is found in the XML database's structure and has its own Trust Value. The write process is limited based on the XML database's access permission policy file that is described in Section 8.3.
- Delete non-existent node: this rule means that the user wanted to delete a node that did not exist in the XML database.

The error rules focus on accessing (read-write-delete) non-existent nodes. Like the bad transaction rules, they can easily be extended to cover other policies. Furthermore, although these rules do not depend on the

existence of a schema – a fixed structure for the XML document – they could easily be extended to cover problems that affect the XML file structure when there is a schema.

The error policy file is a standard XML file that uses some specific tags to define what an error in the system is. The root node is defined by <Errors>. Each error is defined by <Error> and all errors are classified by their identifier and type. The <ID> tag is used to identify each rule. The <Type> defines the rule. The structure of the error policy file is described in Figure 7.2.

```
<Errors>
  <Error >
    <ID> 1 </ID>
<Type> Read nonexistent node</Type >
  </Error>
  <Error >
    <ID> 2 </ID>
 <Type> Write nonexistent node </Type >
  </Error>
  <Error >
    <ID> 3 </ID>
 <Type> Delete nonexistent node</Type >
  </Error>
</Errors >
```

Figure 7.2. The error policy file

7.5 The Operation Recorder

The operation recorder is also a component in the trust module. It aims to store bad transactions and errors for users and provide these recorded operations to the trust calculator. After the operation evaluator and the error detector capture bad transactions and errors, the operation recorder registers these bad transactions and errors in the XLog file. This XLog file is used by the trust calculator to calculate the new Trust Value.

Logging is an important process in databases and is used for recovery and security purposes. Logging in XML databases has rarely been discussed in the literature. The main purpose of logging in normal databases is to record transaction information that is used for recovery when the system crashes and sometimes for concurrency control (Elmasri and Navathe, 2007; H. Molina et al., 2009); it can also be useful for security purposes to track malicious transactions in databases (Etoh et al., 2010). The main classifications in logging are: undo logging, redo logging, and undo/redo logging, all of which are used mainly to restore data (Elmasri and Navathe, 2007; H. Molina et al., 2009). Logs can be represented as tables in databases or files. Log files can be written in various syntaxes, formats, and languages. Wang et al. (2006) suggest that using XML to create the log file saves both time and space compared to tables.

In this system the XLog file is used, not for recovery, but to calculate user Trust Values by temporarily recording users' bad transactions and errors. It is described in the following Sections.

7.5.1 The XLog File

The XLog file for XML databases with Trust Based Access Control, unlike conventional log files, is focused on security rather than recovery. Thus the XLog file will:

- Support a secure environment for access control of XML databases.
- Track user operations and behaviour by recording and organising their actions.
- Produce a log file that can be used to calculate a Trust Value that directly affects user access privileges.

It is dynamic and temporary as it is retained only for a certain period of time depending on the organisation's policy, such as a session, a day, or a week. The XLog file is written in XML and is processed as a normal XML file. Its structure differs from a normal log file, since it depends on the user

identifier instead of time and transaction identifier. Using this structure makes capturing user behaviour fast and easy. It does not need to record the time for each transaction because it is irrelevant to the calculation of trust.

In this file, the root node is defined by <Users>. Bad transactions and errors are grouped for each user by using <User>. The <User> tag consists of three kinds of sub elements: <ID>, <Bad Transaction>, and <Error>. The <ID> tag indicates the user identification. The <Bad Transaction> tag refers to the identifier of a specific bad transaction defined in the operation policy file. The < Error> tag indicates the identifier of the error rule that is defined in the error policy file. The structure of the XLog file is shown in Figure 7.3.

```
<Users>
  <User >
    <ID> 30 </ID>
    <Bad Transaction> 1 </Bad Transaction>
    <Bad Transaction> 4 </Bad Transaction>
    <Error> 1 </Error>
    <Error> 3 </Error>
     …
    </User>
</Users>
```

Figure 7.3 The XLog file

7.5.2 XLog File Features

As mentioned above, the main purpose of the XLog file is to record user behaviour. The features of this file are discussed in this section. The majority of its advantages appear through applying a simple structure and using the XML to write the XLog file. Consequently, the XLog file adopts the advantages of XML such as flexibility and simplicity (Ray, 2003; W3C, 2010; W3Schools, 2013a). The important features are discussed below.

- **Temporary:** the XLog file is created to be used for a certain period depending on the organisation's needs. The organisation and the system administrator can define how long the XLog file will exist. The period

could be a session, a day, or a week. After using data from the XLog file to calculate users' trust by the trust calculator and update their privileges by the access manager, the XLog file is destroyed. This leads to another feature; the XLog file consumes little storage.

- **Dynamic:** one of the main features of the XLog file is that it is dynamic and updated regularly. It reflects misuse as soon as it occurs. This feature is derived from its transient nature. The XLog file is temporary; it is amended to record each fresh transaction and thus contains all recent data.
- **Consumes little storage:** as a result of its temporary and dynamic structure, the XLog file contains only recent processes. Furthermore, this storage is only retained for a defined period.
- **Flexible:** the XLog file is flexible in that it is written in XML. XML gives the users the freedom to create their own tags according as necessary. Even the XLog file structure, defined in the previous Section, can be changed by the administrator and tags can be amended to serve particular needs.
- **Simple:** the XLog file is created to be simple. Through using XML, the XLog file becomes easy to use and understandable for both humans and machines.
- **Interrelated with other files:** the XLog file can be related easily and smoothly with the operation policy file and the error policy file. The content refers to other files by using reference identification <ID>. The organisation can extend errors and bad operations types in the policy files and these can be automatically related to and recorded in the XLog file.
- **Consistent environment:** since the motivation of creating the XLog file is to serve XML databases' security, the XLog file is obviously best written in XML.

7.6 The Trust Calculator

The trust calculator is one of the principal parts in the trust module. It calculates the new Trust Value for the users. It finds both bad transactions and errors recorded in the XLog file and depends on the trust policy file to make the calculations. After calculating the new Trust Value, this trust calculator integrates with the access decision maker to update the Trust Value for the users.

The majority of trust-based access control models define their policy to calculate Trust Value according to their system's needs. They specify which factors are considered in their calculations and then define the rules and equations (see Chapter 4). The calculation process and the structure of the trust policy file for this system are described in detail in the next Sections.

7.6.1 Calculating Trust Values

A new Trust Value (TV) is a float value in the range [0, 1]. 0 denotes the lowest value of trustworthiness and 1 refers to the highest value. The new Trust Value (TV) is generated using three values: Existing Trust Value (ETV), Bad Transaction Factor (BTF), and Error Factor (EF). Each factor is multiplied by a weight that reflects the importance of the factor in the system and shows to what extent the factor affects the final Trust Value (TV). Each weight is a percentage that shows how much the factor will affect the general equation and the new Trust Value (TV).

The weights are Existing Trust Value Weight (ETVW), Bad Transaction Factor Weight (BTFW), and Error Factor Weight (EFW). All ETVW, BTFW, and EFW percentages can be set in line with the organisation's policies. For example, if the organisation does not consider the Error Factor important then EFW can be 1% but if the organisation considers the error rate to be important then EFW it could be 10%.

Both Bad Transaction Factor Weight (BTFW) and Error Factor Weight (EFW) range between 1% and 10%. The Existing Trust Value Weight (ETVW) range is between 80% and 98%. Range values are selected to keep the Trust Value (TV) within suitable bounds. The maximum for both bad transaction and error weights is 10% and not higher because the aim of the system is to adjust user privilege according to behaviour and not to block user access completely. For example, if this weight much higher say as 50%, then the Trust Value (TV) would drop suddenly and dramatically and may cause other access problems. The Existing Trust Value Weight (ETVW) is regarded as the basic value to calculate the new TV. The new TV is derived from the existing one and this explains why its weight should be high. In the practice the range between 80% and 98% was used. This was arrived at by series of experiments.

The trust calculator completes the calculation process for the new Trust Value in a number of steps:

- Find the number of bad transactions and the number of errors recorded in the XLog file.
- Assign values to both the Bad Transaction Factor and the Error Factor depending on the trust policy file.
- Assign weights to each factor of ETVW, BTFW, and EFW depending on the trust policies.
- Calculate the new Trust Value according to the equations that are recorded in the trust policy file.
- Send the new Trust Value to the access decision maker to update the users' privileges.

The value of Bad Transaction Factor (BTF) depends on the Bad Transaction Number (BTNum) found in the XLog file. After the BTNum has been counted, the range for the BTF is selected. There are five ranges:

negligible, low, moderate, high, and extreme. These ranges are defined according to the organisation's policy for classifying bad transactions.

For the purposes of these experiments in Trust Based Access Control for XML databases, the ranges are defined as follows. The negligible range is used when there are no bad transactions in the XLog file and the BTF will be 0. The low range is selected when the BTNum is between 1 and 5 bad transactions. Then BTF will be equal to 0.25. If the number of bad transactions is between 6 and 10, the moderate range is selected and the BTF will be 0.50. The high range reflects a BTNum between 11 and 15 and the BTF will be 0.75. The BTF reaches 1 when the BTNum fits into the extreme range that is defined to be larger than 15. Table 7.1 shows these ranges and the equivalent BTF.

Table 7.1 The equivalent range for the Bad Transaction Number (BTNum)

Range Name	Bad Transaction Number (BTNum)	Bad Transaction Factor (BTF)
Negligible	0	0
Low	0< BTNum< = 5	0.25
Moderate	5< BTNum< =10	0.50
High	10< BTNum < =15	0.75
Extreme	15< BTNum	1

Like the Bad Transaction Factor (BTF), the Error Factor (EF) is defined according to the Error Number (ENum) counted in the XLog file which leads to a range. The EF ranges are also negligible, low, moderate, high, and extreme. Each one reflects how many errors are detected in the XLog file. These ranges can be defined according to the organisation's policy for classifying errors and are shown in Table 7.2.

Table 7.2 The equivalent range for the Error Number (ENum)

Range Name	Error Number (ENum)	Error Factor (EF)
Negligible	0	0
Low	0< ENum< = 5	0.25
Moderate	5< ENum< =10	0.50
High	10< ENum < =15	0.75
Extreme	15< ENum	1

After the trust calculator finds the BTF and the EF, the Trust Value (TV) is calculated. The TV increases when there are no bad transactions or errors but drops markedly when the Bad Transaction Factor (BTF), Error Factor (EF), or both increase. There are four different equations to calculate the Trust Value and each one applies to specific cases. Trust Value equations are:

$$TV = \begin{cases} ETV * ETVW + (1 - BTF) * BTFW + (1 - EF) * EFW, & BTF = 0 \text{ and } EF = 0 \quad (1) \\ ETV * ETVW - BTF * BTFW, & BTF > 0 \text{ and } EF = 0 \quad (2) \\ ETV * ETVW - EF * EFW, & BTF = 0 \text{ and } EF > 0 \quad (3) \\ ETV * ETVW - BTF * BTFW - EF * EFW, & BTF > 0 \text{ and } EF > 0 \quad (4) \end{cases}$$

Equation 1 is used to calculate Trust Value (TV) when there are no errors or bad transactions and Trust Value increases slightly. In this specific case, the three weights must sum to 1. This equation could be simplified, since it is used when the error and bad transaction factors are zero. It can be shown in the following simple form:

$$TV = ETV * ETVW + BTFW + EFW, \qquad BTF = 0 \text{ and } EF = 0 \ (1).$$

If there are errors or bad transactions, (2) or (3) is used to calculate the TV. In general, if there are bad transactions or errors the TV will decrease. As a consequence, the Bad Transaction Factor (BTF) is subtracted from the Existing Trust Value (ETV) in (2). The same principle applies to (3) when

there are only errors without bad transactions; the Error Factor (EF) is subtracted from Existing Trust Value (ETV). Equation 4 is used when there are both bad transactions and errors. It subtracts both the Bad Transaction Factor (BTF) and the Error Factor (EF) from Existing Trust Value (ETV) to find the new Trust Value (TV).

Table 7.3 illustrates some examples of the calculation of a new Trust Value for some general cases when the Existing Trust Value (ETV) = 0.5. More examples and calculation case studies on the Trust Value will be mentioned in the experimental part of the trust module in Chapter 10. The next Section describes the trust policy file.

Table 7.3 The calculation of Trust Value (TV)

ETV	ETVW	EF	EFW	BTF	BTFW	TV
0.5	85%	0	5%	0	10%	0.575
0.5	85%	0.25	5%	0.25	10%	0.387
0.5	85%	0.5	5%	0.5	10%	0.350
0.5	85%	0.75	5%	0.75	10%	0.312
0.5	85%	1	5%	1	10%	0.275

7.6.2 The Trust Policy File

The trust policy file contains all policies that are related to the calculation of the Trust Value. These policies can be classified as the Bad Transaction Factor policy, the Error Factor policy, the weights policy, and the equations policy. The bad transactions policy is used to assign a value for BTF based on the number of errors recorded in the XLog file. The Error Factor (EF) policy aims to find the value for the EF using the number of errors in the XLog file. Both the Bad Transaction Factor policy and Error Factor policy were described in the previous Section (7.6.1). The weights policy stores the percentages of ETVW, BTFW, and EFW. These percentages are defined by the administrator according to the organisation strategies. The

equations policy contains the equations that are used to calculate the new Trust Value. These equations were also explained earlier in Section 7.6.1.

Like other policy files, the trust policy file is a standard XML file. It starts with the root node <New Trust Value>, which has four children. Each of these sub nodes represents one of the four policies. The Bad Transaction Factor policy is defined by <Bad Transaction Factor>. The Error Factor policy is defined by <Error Factor>. <Weights> is used to represent the weights policy. This node consists of three sub elements: <ETVW>, <BTFW>, and <EFW>. Each one of these elements contains a percentage for the specific factor. The equations policy is defined by <Equations>. This tag contains the equations syntax to calculate the new Trust Value. The structure of the trust policy file is shown in Figure 7.4.

```
<New Trust Value >
<Bad Transaction Factor >
IF BTNUM=0    Then BTF=0, “Negligible”.
IF 0<BTNUM<=5    Then BTF=0.25, “Low”.
IF 5<BTNUM<=10    Then BTF=0.50, “Moderate”.
IF 10<BTNUM<=15    Then BTF=0.75, “High”.
IF 15 <BTNUM     Then BTF=1, “Extreme”.
</Bad Transaction Factor >
<Error Factor>
IF ENUM=0    Then EF=0, “Negligible”.
IF 0<ENUM<=5    Then EF=0.25, “Low”.
IF 5<ENUM<=10    Then EF=0.50, “Moderate”.
IF 10<ENUM<=15    Then EF=0.75, “High”.
IF 15 <ENUM     Then EF=1, “Extreme”.
</Error Factor>
<Weights>
<ETVW>85%</ETVW>
<BTFW>10%</BTFW>
<EFW>5%</EFW>
</Weights>
<Equations>
Where EF=0 and BTF=0 Then TV= ETV*ETVW + (1-BTF)*BTFW + (1-EF)*EFW.
Where EF=0 and BTF>0 Then TV= ETV*ETVW - BTF *BTFW.
Where EF>0 and BTF=0 Then TV= ETV*ETVW - EF *EFW.
Where EF>0 and BTF>0 Then TV= ETV*ETVW - EF*EFW - BTF*BTFW.
</Equations>
</New Trust Value>
```

Figure 7.4 The trust policy file

7.7 Conclusion

The trust module is the first part of the system. It is integrated with the access control module to perform the system's process. The trust module aims to capture the user behaviour and calculate the Trust Value. It consists of four components and all of them work together. The components' functions depend on the policy files. Defining specific policies is difficult because the rules will be different from system to system and from one organisation to another. Therefore, policies are defined in general and can be changed or extended according to system and organisation needs. The second part of the system, which is the access control module, is described in the following Chapter.

8 THE ACCESS CONTROL MODULE

8.1 Introduction

The Trust Based Access Control for XML databases consists of two modules: the trust module and the access control module. The trust module was described in detail in the previous Chapter. In this Chapter the access control module is explained. This module consists of two parts: the access manager and the access decision maker. Section 8.2 gives the overall view of the access control module. The access manager is explained and the access permission policies for both users and data are described in Section 8.3. The access decision maker is described in 8.4. The conclusions are summarised in Section 8.5.

8.2 Access Control Module Overall

The access control module is connected to the trust module (see Chapter 6), which is the other important part of the Trust Based Access Control system. The combination makes the access processes dynamic and responsive to the current evaluation of users' Trust Values. The access control module:

- Stores the access rules for both users and data in the policy files.
- Checks the queries and makes the decision whether access is to be permitted or denied.
- Searches and retrieves data from the XML database when access is approved.
- Update users' privileges depending on the Trust Values that are provided by the trust module.

The access process in this module depends on the access manager and the access decision maker. The access manager deals with access policies; the access

decision maker determines, in the light of these policies, whether the access can be permitted or denied. Both the access manager and the access decision maker mechanisms are explained in the following Sections.

8.3 Access Manager

The main goal of the access manager is to store and handle permission policies, which are dependent on Trust Value. The access manager breaks down policies into two parts: subject policy related to the user and object policy related to the data. Subject policy uses the user's identification, role, and trust factor. The identification is a serial number to distinguish users easily. The role is similar to 'role' in conventional access modules but in this system is far less significant. The role is given to the user in the initial stage. The Trust Value (TV) is dynamic and updated according to an evaluation of users' behaviour over time. It is used to capture and prevent misuse by all users but especially from insiders who exploit their role and take advantage of their position. The initial TV for each user is assigned by the administrator and then it is changed, as described, according to users' behaviour over time.

The subject policy is recorded in a users' access permission policy file. In this file, each user is assigned a Trust Value according to his history of errors and bad transactions. The variable TV provides different access permissions for the same role. For instance, the managers in the system can have different TVs from each other. At the same time, the TV is amended inside a specific range that is defined by using boundary management. As mentioned in Section 6.6, there are two boundaries: the maximum and the minimum boundary. The maximum boundary is the highest TV that the user can reach. The minimum boundary is the lowest TV. The idea of using this boundary is to change the access permission for users according to their behaviour and, at the same time, ensure the access permission is not blocked completely. For example, the highest boundary for staff is 0.75 and the minimum boundary is 0.5. Subsequently, the

TV for each member of staff can be increased or decreased inside this defined range.

The users' access permission policy file is written in XML and is handled as a standard XML file. Some specific tags are used to define the access permission for users. The file starts by the root node <Users> that includes all users in the system. The access privilege for each user is defined by using <User>, which consists of three sub elements: <ID>, <Role>, and <TV>. The <ID> refers to the user identification. The <Role> reflects the user role in the system. The role is recorded to support the boundary management that were described in Chapter 6. <TV> is the Trust Value for the user that manages the user's privileges to access data. Figure 8.1 shows the structure of the users' access permission file.

```
<Users>
<User>
<ID> 1 </ID>
<Role> manager </Role>
<TV> 0.8 </TV>
</User>
<User>
<ID> 57 </ID>
<Role> staff </Role>
<TV> 0. 50 </TV>
</User>
<User>
. . .
</User>
. . .
. . .
</Users>
```

Figure 8.1 The users' access permission policy file

Like the subject policy, the data policy assigns a Trust Value to each item of data. A high Trust Value reflects that the data are sensitive and need higher levels of protections and vice versa. Each node has its own TV independent of other nodes. The required node is accessed through XPath to ensure that no

inappropriate node will face disclosure. For consistency, the parent node has the lowest TV that is assigned to any of its children. Compared to subject policy, data policy is far more static and rarely needs to change. The administrator is responsible for assigning the appropriate TV for each node. Using this strategy explains why the user cannot add new nodes that do not exist in the object policy. As mentioned in Chapter 7, this process is recorded as an error because when a normal user adds a new node that is not defined in the system and does not have a TV, he cannot assign the appropriate TV for it since that is the administrator's responsibility. The administrator can add new nodes to the structure in XML files and assign the appropriate Trust Value for each node.

The object policy is recorded in an XML database's access permission policy file and is dealt with as a normal XML file. The file contains all nodes in the original XML database, but it is relatively small because it includes the nodes without repetition. Each node has the proper TV according to its sensitivity and importance. The file starts with the root node <Database> that includes all nodes. The TV for each node is a content of the <Node> element. The structure of this file is explained in Figure 8.2.

```
<Database>
<Node1 > 0.5
 <Node2> 0.5 <Node2>
 <Node3> 0.75 </Node3>
</Node1>
<Node4> 0.25 </Node4>
 . . .
 . . .
</Database>
```

Figure 8.2 The XML database's access permission policy file

8.4 Access Decision Maker

The access decision maker is part of the access control module. It analyses the XML query and checks policies in the access manager to reach the

final decision whether to permit or deny the request. This process is carried out in a number of steps:

- Search the Trust Value for the user in the users' access permission policy file.
- Search the Trust Value for the queried data in the XML databases' access permission policy file.
- Compare the user's Trust Value and data Trust Value. If the user's Trust Value equals or is larger than the data Trust Value and the process is not classified as a bad transaction or error then the user can access the data in the XML database; otherwise, access is denied.
- Search and retrieve the required data from the XML database when the access is permitted. The XML database is stored natively in the system and is represented as a tree when the access is allowed. Accessing the required node is achieved using an XPath expression.

The access decision is dependent on Trust Values. It is connected with the access manager to determine the Trust Value (TV) for both users and data. The access decision maker also works coherently with the trust module to check the queries entered and decide whether access is permitted or denied. Figure 8.3 shows the decision process in the access module for XML databases.

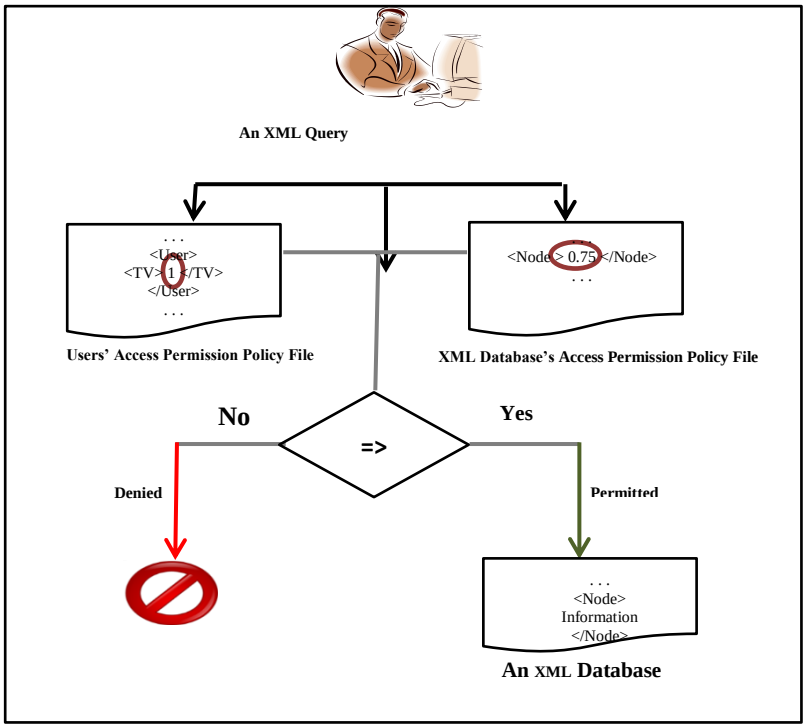


Figure 8.3 The access process in the access control module

8.5 Conclusion

The access control module is part of the Trust Based Access Control for XML databases. It is responsible for storing access permission policies and makes the decision whether access is allowed or not. It includes two parts – the access manger and the access decision maker – that work together to achieve the main goal of this module. This module is integrated with the trust module to perform access control. In the next Chapter, the design of experiments to test this approach is explained and the data sets are described.

9 THE EXPERIMENTAL DESIGN

9.1 Introduction

Chapters 6 to 8 explained the structure of Trust Based Access Control for XML databases. This Chapter explains the design of experiments used to evaluate the system performance, functionality, scalability, and security.

Seven different experiments were run to test the system units, the system integration, and the whole system. The first three experiments evaluate specific parts of the system individually: the trust module, the access control module, and the XLog file. Two other experiments evaluate the performance of the system's processes, namely the trust maintenance and the access supervision (see Chapter 6). The last two are comparison experiments. The sixth experiment measured the real time cost of applying Trust Based Access Control (TBAC) by comparing the proposed system with another system that does not depend on TBAC. The final experiment compares the proposed system with the traditional Mandatory Access Control (MAC).

The remainder of this Chapter is divided as follows. Section 9.2 describes the objectives of each experiment and the general strategy for evaluation. Section 9.3 explains the tool considerations and the platform specifications. In Section 9.4, the review for existing data sets and XML benchmarks and the selection of data sets' criteria are made. The samples of users are mentioned in Section 9.5. Section 7.6 explains the setup for each experiment. A summary is provided in Section 9.7.

9.2 The Overall Experimental Design

The main goal for running the test described here is to evaluate the research hypothesis stated in Chapter 5. Experiments were designed to ensure that the system design described in Chapters 6-8 met the system objectives and requirements mentioned in Chapter 5. To evaluate the validity, functionality, performance, and security of Trust Based Access Control for XML databases, seven different experiments were run. The early stage of development of this work is not yet at a phase where rigorous hypotheses could be developed for the experiments, as would be required for formal experiments in empirical software engineering. So, these experiments were designed only for the general scientific sense and not for the more specialised sense used in empirical software engineering.

The system was tested practically using three levels: unit testing, integration testing, and system testing. Three experiments were designed to test system units, two experiments were used to evaluate the integration between modules; a further two experiments were run to check how the whole system worked and compares with other existing approaches. These experiments are:

- The trust module experiment
- The logging experiment
- The access control module experiment
- The trust maintenance experiment
- The access supervision experiment
- The experiment to determine the cost of Trust Based Access Control
- The comparison with MAC experiment

All these experiments and their objectives are described in detail in the following Sections.

9.2.1 The Objectives of The Experiments

As mentioned before, these experiments aim to evaluate applying Trust Based Access Control for XML databases. They check the system performance, security, and functionality. The system's main functions were mentioned in the design description in Chapters 6-8; they can be summarised as follows:

- Evaluating user operations by checking XML queries, capturing errors, and bad transactions.
- Recording errors and bad transactions.
- Calculating the Trust Value (TV).
- Making the access decision for XML databases.
- Retrieving data from XML databases.
- Updating user's privileges according to their behaviour.

Each experiment is designed to test some tasks in the system and achieve specific goals. The objectives of each experiment are described below.

- **The trust module experiment:** This experiment aims to test the trust module in the system as an individual unit. The main goal of the trust module experiment is to identify appropriate ranges of values for the various factors and weights. It also evaluates the Trust Value performance. Trust Value is changed depending on the Existing Trust Value (ETV), Error Factor (EF), Bad Transaction Factor (BTF), and their weights. This experiment tests how the Trust Value is affected by these factors from two perspectives. The first viewpoint shows the change in Trust Value depending on the existing Trust Value, Error, and Bad Transaction Factors. The second explains how the weight values affect calculation of the Trust Value. The expected results from these experiments are that TV increases when there are no errors or bad transactions and decreases when there are only errors, bad transactions, or both. The results will be linear.

- **The logging experiment**: This experiment evaluates the performance of the XLog file for XML databases. It checks its performance speed over time from two perspectives. The first is to test the creation process. The second focuses on the reading process and retrieval of data. Both perspectives are evaluated in three ways: with errors only, bad transactions only, and a mixture of both errors and bad transactions. The expectations for results are that both reading and creation processes will consume little time due to the design of the XLog file being dynamic and updated.

- **The access control module experiment:** This experiment is designed to evaluate the functionality, performance, and scalability of the access control module. It evaluates each step of the access process and also tests the whole access module. From scalability perspective, the access module works with small to large databases. The data sets and their size are explained in Section 9.4. Similarly, the access module is tested with different sized users' access permission policy files reflecting different number of users. The selection of the number of users will be described in Section 9.5. The expected results are that the access process in this module will consume much time since it includes a searching process for TV for both users and data.

- **The trust maintenance experiment:** This experiment aims to test the integration between the trust module and the access control module. In this experiment, the performance of both modules working together is tested. It evaluates the trust maintenance process, described in Chapter 6, with different numbers of users, errors, and bad transactions. The total time to perform this process is measured. This time includes the evaluation time for errors and bad transactions recorded in the XLog file, the calculation time, and the time to update privileges, which are recorded in the user's access permission policy file. The performance of this process is expected to be reasonable, due to the XLog file design.

- **The access supervision experiment**: Like the trust maintenance experiment, this experiment evaluates the integration of two modules. It tests the performance of the access supervision process, which is described in Chapter 6, by evaluating the time needed to finish this process. The time includes the access time, the time for detecting errors and bad transactions, and then the time required for recording in the XLog file. The expected completion time for this process when the access is permitted will be longer than when it is denied because it excludes the retrieval time from the original XML databases. Both situations will take long time since the access supervision process includes three complex sub processes: accessing, detecting, and creating the XLog.

- **The experiment to determine the cost of TBAC:** This experiment aims to test the whole system performance and scalability. It evaluates the time needed to apply the Trust Based Access Control for XML databases by comparing the time consumed with and without the Trust Based Access Control. It also checks the scalability of the system by using small to large XML databases. The size of data sets is explained in Section 9.4. The expected results are that TBAC will be applied successfully and will add more security features while take four times the normal access time.

- **The comparison with Mandatory Access Control (MAC) experiment:** This experiment aims to evaluate the performance of Trust Based Access Control for XML databases through comparing it with other existing approaches. The Mandatory Access Control for XML databases designed by (Zhu et al., 2009) is selected because it is a traditional access type. It has been implemented practically and has published experimental results. This experiment measured the time consumed for both Trust Based Access Control and Mandatory Access Control for several sizes of XML databases. The expected results are that the performance of TBAC will be slower than

traditional MAC performance, but at the same time it will add dynamic security features.

9.2.2 The Strategy of The Experimental Evaluation

To achieve all objectives mentioned in above, the design of experiments must cover these points:

- The operational environment for these experiments. Both software and hardware are identified according to experiments' aims (see Section 9.3).
- The choice of XML data sets for each experiment and the number of user (see Sections 9.4 and 9.5).
- The choice of the queries depending on the objectives of experiments.
- The inputs and outputs for each experiment and the unit of measurement.
- Analyses of the experimental results (see Chapter 10) and evaluations of the system (see Chapter 11). Identifying the significant results and possible improvements (see Chapter 12). The next Sections and Chapters describe these points in detail.

9.3 The Implementation Platforms and Tools' Considerations

Practical Trust Based Access Control for XML databases was tested on two platforms. The five experiments that focus on unit testing or integration testing were performed on a laptop with 2.40 GHz Intel® Core™ i5 CPU, 4 GB of main memory, and Windows 7 operating system.

The second platform is a PC with 2.83 GHz Intel® Core™ 2 Duo CPU E8300. The system is implemented using Java Language (JDK 1.7.0) and Net Beans IDE 7.0.1 platform framework. Several XML databases with different file sizes and different numbers of users are used in experiments. The detail of the data sets and users appear later in Sections 9.4 and 9.5.

The further two experiments that focus on comparison were performed on platform two. The specifications of platform two were selected to be the same

as in the practical experiments of the Mandatory Access Control approach (MAC) (Zhu et al., 2007; Zhu et al., 2009) to obtain better and more accurate comparative results.

The practical Trust Based Access Control for XML databases adopts DOM as a parser over SAX. As mentioned in Chapter 2, DOM parses the XML file and represents it as a tree. It supports XPath and Queries. It provides easy navigation and traversal in any direction. It allows reading and manipulation of data. The only problem with DOM is that it consumes memory space. SAX, on the other hand, is simple and offers high performance. It creates a stream and represents the elements as events. It supports only up-down traversal and reading without manipulation.

The main reason for selecting DOM as a parser for this system was to meet the system objectives and requirements. The system needs to represent the XML file as a tree and understands the whole structure to capture specific kinds of bad transactions and errors. For example, to check if the required node is a root or a parent of children. Moreover, manipulating XML data is an important process in the system to evaluate the user transactions. Another logical reason is that DOM supports XPath, which is used in the practical system to access XML data. Although SAX is faster, DOM provides more functions. However, using DOM may restrict the size of data sets that used in the experiments. As mentioned in Section 2.6, XPath was selected over XQuery to access data due to its simplicity. The next Section reviews existing data sets and describes the selection criteria for each experiment.

9.4 Real-World Data Sets and XML Benchmarks

This Section discusses some real XML data sets and benchmarks. The discussion briefly covers the properties of each data set and benchmark in Sections 9.4.1 and 9.4.2. A real XML data set is a single XML file that includes real data. An XML benchmark is a tool that generates synthetic data sets with different sizes and provides query sets. Both real data sets and benchmarks are

used to evaluate the system performance. Since benchmarks provide data sets and query sets at the same time, the storage techniques and query processing can be easily compared between the experimental approach and other existing systems (Schmidt et al., 2001). The choice of real-world data sets and the benchmarks that are used in the experiments is explained in Section 9.4.3.

9.4.1 A Review of Existing Real-World Data Sets

These data sets include real data and structures that make the evaluation process simpler than synthetic benchmarks data sets (see Section 9.4.2) (Mlynkova, 2008). Six real data sets, which are the most widely used in XML evaluations, are described here. All these data sets are free and can be download from the XML Data Repository website (Suciu, 2002).

- **DBLP Database:** This data set is an acronym for Digital Bibliography Library Project. It is a large XML file that includes genuine bibliographic information about computer science publications. These publications cover the major conferences (e.g. VLDB, PODS, ICDE), journals (e.g. CACM, TODS, TOIS), series (e.g. LNCS/LNAI, IFIP), and books in the field of computer science (Suciu, 2002; DBLP, 2013). Many applications for XML databases (Liefke and Suciu, 2000; Wang et al., 2003; Lawrence, 2004; Lu et al., 2005; Xu and Papakonstantinou 2005; Chen et al., 2006; Li et al., 2007; Al-Badawi, 2010) used this data set in their evaluation experiments. This data set has a simple, shallow, and wide structure (Lu et al., 2005; Chen et al., 2006; Lee et al., 2010). The original version of this data set can be downloaded from the DBLP website (DBLP, 2013). The size of this data set is extremely large. On 14th March 2013, the size of the DBLP database was around 1.1 GB (DBLP, 2013). Some features of the smaller version with 127 MB of this data set are provided in Table 9.1 (Suciu, 2002).

- **Protein Sequence Database:** This database is designed by Georgetown University. It is a resource of integrated bioinformatics that includes information about the protein sequence. Like DBLP, this data set is a large XML file and has a shallow, wide, and regular structure (Wong et al., 2007). The size reaches 683MB and the depth is seven levels. It has been used to evaluate experiments on the XML storage (Wong et al., 2007), the processing XML streams (Green et al., 2003; Jittrawong and Wong, 2007), and filtering (Suciu, 2002; Silvasti et al., 2009).

- **Treebank Database:** This database was developed by the Computer and Information Science Department at the University of Pennsylvania. It includes English sentences that are annotated for linguistic structures. This database is partially encrypted to protect copyright for text nodes. This encryption does not affect the XML structure at all. This data set is considered an interesting case for evaluation experiments due to its deep recursive structure (Onizuka, 2003; Chen et al., 2005; Lu et al., 2005; Chen et al., 2006; Wong et al., 2007). The tree contains a huge number of nested structures 386,614 and is considered as a complex XML database (Onizuka, 2003). This real data set is widely used to evaluate different aspects of many XML applications (Liefke and Suciu, 2000; Green et al., 2003; Onizuka, 2003; Steedman et al., 2003; Chen et al., 2005; Lu et al., 2005; Chen et al., 2006; Li et al., 2007; Wong et al., 2007). The size of this XML file is 82MB (Treebank, 1999; Suciu, 2002). Basic statistical information about this data set is provided in Table 9.1.

- **NASA Database:** This database contains genuine astronomical data. This XML data set is generated from a flat file format. It is part of the GSFC/NASA XML Project. The size of the XML document is 23 MB (NASA, 2001; Suciu, 2002). Unlike Treebank, this data set is shallow. The number of recursive elements is only 18 (Onizuka, 2003). It is used to test

many XML applications that are designed for processing XPath and XML queries (Green et al., 2003; Onizuka, 2003; Zhang et al., 2005; Jittrawong and Wong, 2007), indexing techniques (He and Yang, 2004), labelling (Wu et al., 2004), filtering (Silvasti et al., 2009), and searching (Lee et al., 2010). Other features of the XML structure are mentioned in Table 9.1 (Suciu, 2002).

- **SIGMOD Record database:** This data set includes real data for some articles published by the ACM SIGMOD website. It is a relatively small database since the XML file size is around 0.5 MB (Merialdo., 1999; Suciu, 2002). This database is generally used to evaluate the XML systems' performance with small XML databases (Li and Moon, 2001; Lawrence, 2004; Wu et al., 2004; Rafiei et al., 2006; Li et al., 2007; Lee et al., 2010). The database structure features are explained in Table 9.1. More detail about this data set can be found on the ACM SIGMOD Record website (Suciu, 2002).

- **University Courses Database**: This data set contains information for courses in three universities. There are three small versions of this database with different sizes. The first version is a small XML file with 277KB and the number of levels is four. The size of the second version is 1MB and the depth is four levels. The third version is 2MB with five levels. Although the number of versions is limited, the different sizes for this data set support scalability tests to some extent (Suciu, 2002).

Table 9.1 Real-world XML databases' features

Database name	**DBLP**	**Protein sequence**	**Treebank**	**NASA**	**SIGMOD Record**	**University courses**		
Database size	127 MB	683 MB	82 MB	23 MB	467 KB ≈ 0.5 MB	277KB	1MB	2MB
The number of nodes	3,736,406	22,596,465	2,437,667	532,963	15,263	10,546	74,557	66,735
The number of elements	3,332,130	21305818	2,437,666	476,646	11,526	10,546	74,557	66,729
The number of attributes	404,276	1290647	1	56,317	3,737	0	0	6
The number of levels	6	7	36	8	6	4	4	5

9.4.2 A Review of Existing XML Benchmarks

XML benchmarks were developed to consider data storage and query processing (Schmidt et al., 2001). XML benchmarks can be classified into application benchmarks or micro benchmarks (Yao et al., 2004; Runapongsa et al., 2006b). The application benchmarks aim to evaluate the performance of the XML database as a whole with both data and queries. In contrast, the micro benchmarks focus on evaluating aspects of a specific component in the system such as query processing (joins, grouping, and sorting) (Yao et al., 2004; Runapongsa et al., 2006b). The most popular and widely used XML benchmarks are discussed in this Section.

- **XMark Benchmark:** This benchmark has the ability to generate several sizes of XML database. The query set is designed to cover most of the query-able aspects. XMark was developed by Schmidt et al. (2002) and is widely used to evaluate XML applications (Davis et al., 2003; Wang et al., 2003; Arion et al., 2004; He and Yang, 2004; Lawrence, 2004; Chen et al., 2005; Lu et al., 2005; Chen et al., 2006; Li et al., 2007; Lee et al., 2010). XMark generates the data set as a single XML file that contains simulated data of an online auction website. An XMark data set is easy to understand. The generator of XMark data set is free to download from the XMark project website (Schmidt, 2003). The size of the database is controlled by a scaling factor. So, it allows developers to generate their data sets according to their needs. The number of levels of the XML tree (depth) is always twelve regardless of the size of XML file. It has a repetitive structure with a fair number of recursions (Chen et al., 2005; Zhang et al., 2005). XMark data sets are an appropriate tool to evaluate the system performance especially from the scalability perspective. Although, this benchmark provides a query set that is designed to evaluate several aspects of databases, this query set does not cover update transactions. There are twenty queries that focus on

searching transactions (Schmidt, 2003). XMark's basic features are summarised in Table 9.2.

- **XOO7 Benchmark:** Li et al. (2001) applied the original idea of Object Oriented RDBMS benchmark (OO7) that was developed by Carey et al. (1993) to the XML environment. The data and query sets of OO7 were converted to be used in the XML version of the benchmark (XOO7). XOO7 also generates an XML data set as a single XML file. There are three versions of this data set: small, medium, and large. The limitations in data sets size restrict the scalability evaluation. The depth of this data set, like that of XMark, is static irrespective of the size. For XOO7 it is five deep with all three versions. The query set includes twenty three queries that cover search processes again without update (Li, 2003). Some features of XOO7 benchmark are explained in Table 9.2. The XOO7 benchmark is freely available on the XOO7 Benchmark website (Li, 2003).

- **XBench Benchmark:** XBench is a template based benchmark that generates a wide design of XML files. It can generate data centric XML files (DC) and text centric XML files (TC). The database can be in a single XML document (SD) or multiple XML documents (MD). Four classes of XML databases can be created by the toXgen tool: DC/SD, DC/MD, TC/SD, and TC/MD. This benchmark provides four XML databases sizes. The small database is 10MB, the normal size reaches 100MB, the large size is 1GB, and the huge size is 10GB (Yao et al., 2004). The benchmark has the same drawbacks as XOO7; the sizes of the databases are fixed. Unlike XMark and XOO7, this benchmark provides a limited range for the number of levels (depth) that is set by parameter. XBench contains twenty queries that focus on search without update.

- **XMach-1 Benchmark:** This benchmark is designed to be multi-user. It was developed by Böhme and Rahm (2003). It depends on a web based application scenario. The structure of this benchmark consists of four parts: the XML database, server, loader, and client. Figure 9.1 shows the architecture of XMach-1 benchmark. The XMach-1 data set consists of a large number of small XML files. There are four versions of the data set size depending on the number of XML files, which can be10^4, 10^5, 10^6, and10^7. The size of each XML file is between 2KB and 100KB. The maximum depth is six levels. The query set contains eleven queries. Eight of them consider the search processes and the other three focus on update transactions (Böhme and Rahm, 2000). Table 9.2 shows some basic characteristics of this benchmark. Both data set and query set for this benchmark are available on the website of XMach-1: A benchmark for XML Data Management (Böhme and Rahm, 2000).

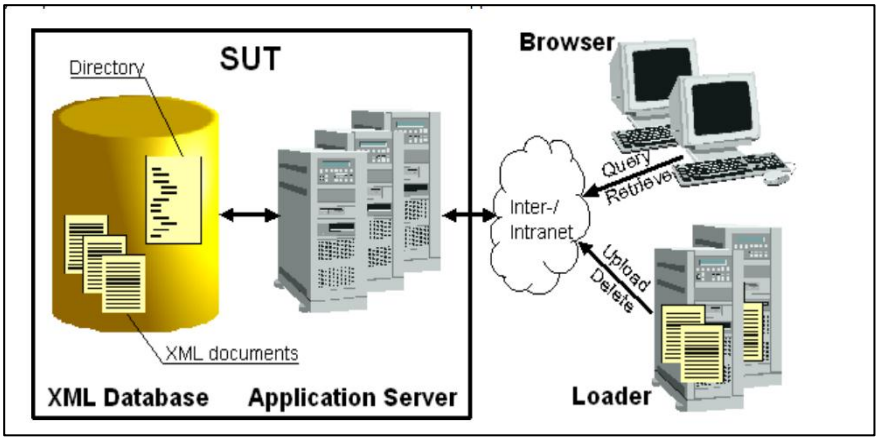


Figure 9.1 The structure of XMach-1 Benchmark (Böhme and Rahm, 2000)

- **The Michigan Benchmark:** This benchmark is classified by its authors as a micro benchmark that is designed to evaluate specific features in the system (Yao et al., 2004; Runapongsa et al., 2006b). It was developed by Runapongsa et al. in (2006b). The data set is generated as a single XML file that contains a minimum of 728,000 nodes. The maximum number of nodes

is ten times as large. The depth of this data set is sixteen levels and the breadth is changeable. The breadth is defined by a fan-out parameter, of which the minimum value is two nodes at each level and the maximum is 13 nodes. The query set contains thirty one queries that are designed to test several aspects of databases including update processes (Runapongsa et al., 2006a). Some characteristics of the Michigan benchmark are mentioned in Table 9.2. This benchmark can be found in the project website (Runapongsa et al., 2006a).

- **TPoX benchmark:** TPoX is an acronym of Transaction Processing over XML. It is an application benchmark that aims to evaluate the whole system. The generation of XML files process depends on templates. XML Schema is used to control the size of XML files by defining the depth and the breadth of the database. The database is generated as multiple tiny XML files. The size of each file is between 2KB and 20KB (Nicola et al., 2007b). This benchmark provides seventeen queries that focus mainly on updating XML databases, unlike other benchmarks that are more concerned with searching processes. Some features of this benchmark are shown in Table 9.2. The benchmark can be downloaded from the project website (Nicola et al., 2007a).

Table 9.2 Features of some XML benchmarks

Benchmark Name		XMark	XOO7	XMach-1	Michigan	XBench	TPoX
Data set	The number of files	1	1	Multiple (10^4, 10^5, 10^6, and 10^7) files	1	Mixed	Multiple From 3.6×106 to 3.6×1011
	The size	Varies From tiny (KB) to huge (GB)	Small (500B), Medium (1000B), Large (1000B) with different nodes number.	From 2 to 100KB per XML file	Min: 728,000 nodes. Max: 10 times Min	Small (10MB) Medium(199MB) Large (1GB) Huge (10 GB)	From 2KB to 25KB per XML file
	The number of levels	12	5	≤ 6	5 to 16	Limited range	Multiple Controlled by template
Query set	The number of queries	20	23	11	31	20	17
The number of users		1	1	Multiple	1	1	Multiple

9.4.3 The Experimental Data Sets' Criteria

In order to test the practical Trust Based Access Control for XML databases, seven different experiments were designed (see Section 9.2). These experiments test the system from different perspectives. The selection of data sets and benchmarks from those described in Sections 9.4.1 and 9.4.2 was made based on the experiments' objectives.

In general, real-world data sets were selected in preference to benchmark data sets for the five experiments: the trust module experiment, the logging experiment, the access control module experiment, the trust maintenance experiment, and the access supervision experiment. This is because real-world data sets are simple but contain realistic data. The benchmarks are generally used to test the management systems of XML databases and focus on storage techniques and query processing. Since this system handles security issues, using natural data sets is more appropriate.

From the data sets reviewed in Section 9.4.1, Treebank, NASA, and SIGMOD Record were used in these five experiments. These three genuine data sets provide an environment to evaluate the system with different database size, structure, and depth. DBLP and Protein Sequence Database were excluded due to the large size of the XML files, which require higher specifications of hardware and software. Treebank was selected because of its complex recursive structure. Lee et al. (2010) state that DBLP, NASA, SIGMOD Record, and XMark are the most popular data sets used to evaluate experiments in research in XML databases. Therefore, NASA and SIGMOD Record were selected in preference to the University Courses database.

The two comparison experiments, the comparison between the system with and without Trust Based Access Control (TBAC) and the comparison between Mandatory Access Control (MAC) and Trust Based Access Control (TBAC), used the XMark benchmark for many reasons. The most important one

is that XMark was used in the practical experiments of the Mandatory Access Control system (MAC) designed by Zhu et al. (2009). The comparison experiments must run with the same environment including benchmarks to obtain accurate performance results. XMark is an appropriate tool to evaluate the whole system and compare different approaches. XMark is a well-known benchmark that provides XML databases in several sizes that make the scalability tests easier.

Eleven XML data sets were generated to compare the Trust Based Access Control with other systems. The minimum size was 27KB for XFile1 and the size gradually increased for other files. The size 30.2MB in XFile11 was the maximum due to the limitations in the resources. Moreover, using DOM as a parser in the system consumes much memory storage. Table 9.3 shows the XML files generated by XMark and used in the comparison experiments.

Table 9.3 The size of XMark data set used in comparison experiments

File Name	Scaling Factor (F)	File Size (MB)
XFile1	0.00001	0.027
XFile2	0.005	0.56
XFile3	0.01	1.15
XFile4	0.02	2.3
XFile5	0.05	5.7
XFile6	0.061	7
XFile7	0.087	10.1
XFile8	0.13	15
XFile9	0.175	20.2
XFile10	0.22	25.5
XFile11	0.26	30.2

In contrast, XOO7 and XBench benchmarks provide only restricted XML databases' sizes and do limit the scalability evaluation in comparison experiments. The XMach-1 and TPoX benchmarks are also not useful because the database is divided into many XML files. The experimental system requires an understanding of the whole database structure and all data to capture errors and bad transactions. The Michigan benchmark is a micro benchmark, thus it is not an appropriate tool to test this system either.

9.5 The Sample of Users

This Section explains the user samples employed in the experiments. The selection of the number of users was made to cover small, medium, and large organisations. The size definition for companies varies from country to country. The Companies Act (2006) of the United Kingdom and defines legislations for different sized organisations. These relate to several points such as annual sales (turnover), balance, and number of employees. Only the number of employees is relevant for this research. According to the Companies Act (2006), the maximum number of employees in a small company is 50. For the medium sized organisation, the number of employees is not more than 250. If the number of users is larger than 250 the company is classified as large. Furthermore, the European Union defines the business sizes similarly to the definition of the United Kingdom. A large company has no more than 1,000 employees. When the number of employees is greater than 1,000 the company is considered an enterprise.

The United States defines the number of employees for small and medium organisations differently. The maximum number of employees is 250 for a small company and 500 for a medium company, while the same numbers as in the European Union apply for large and enterprise companies. In a large company, the number of employees is greater than 500 and less than or equal to 1,000. An enterprise company has more than 1,000 employees.

The business size definition depending on the number of employees is completely different in Australia. The small company according to the Fair Work Act (2009) includes no more than 15 employees. The number of employees in the medium company is less than or equal to 200. The maximum number of employees in the large company is 500. If the number of employees is larger than 500, the company is categorised as an enterprise.

Since this research takes place in the United Kingdom, it follows the organisation size definition produced by the government of the United Kingdom. Thus the selection for the number of users was 50, 100, and 1,000, which covers small, medium, and large organisations.

The data for the users were virtual and not real. Only information related to user privileges are recorded in the users' access permission policy file. The file structure was described in Chapter 8. The file contains the ID, Role, and Trust Value (TV) for each user. ID refers to the user's identification. Role describes the user's role in the organisation. TV reflects the trustworthy value for the user according to the behaviour.

Three versions of a users' access permission policy file were generated with 50, 100, and 1,000 users.

These user sets are used to test the system in all designed experiments expect the trust module experiment, because that experiment cannot be affected by the number of users. The next Section describes the input required for all experiments and how they are run.

9.6 The Setup of Experiments

As mentioned earlier, seven different experiments were designed to evaluate the Trust Based Access Control for XML databases. The objectives of

each experiment were explained in Section 9.2. In this Section, the setup of experiments is described.

9.6.1 The Trust Module Experiment

This experiment is used to find the appropriate values for factors and their weights. It evaluates performance of calculation of the Trust Value (TV). It shows various situations for calculating TV depending on Existing Trust Value (ETV), Bad Transaction Factor (BTF), Error Factor (EF), Existing Trust Value Weight (ETVW), Bad Transaction Factor Weight (BTFW), and Error Factor Weight (EFW).

The procedure for running this experiment is that a new Trust Value (TV) is calculated using formulae in Chapter 7 from values: ETV, BTF, EF, ETVW, BTFW, and EFW. The initial values are varied, keeping some stable, to check the effect on the output. Thus the experiment studies the relationship between TV and these six factors.

9.6.2 The Logging Experiment

This experiment is divided into two parts. The first part focuses on evaluating the creation process for the XLog file. The second part tests the performance of retrieving data from the XLog file. Both write and read processes are compared using thirty XLog files created with different sizes according to the type and the number of recorded processes. Three types of XLog file are defined: an XLog file with only bad transactions, an XLog file with only errors, and the XLog file with both bad transactions and errors. The first ten versions of the XLog file include only bad transactions and the number of bad transactions increase by ten for every version. The second ten versions of the XLog file contain only errors and also increased by ten errors each time. The third type is XLog files include a mixture of both bad transaction and errors. They too increase by ten bad transactions and ten errors each time.

9.6.2.1 Creating the XLog File

The creation process includes generating the XLog file and writing detected bad transactions and errors. The technical setup for the creation process of the XLog file is summarised here. The time consumed to create the XLog file is measured in milliseconds. This procedure was repeated thirty times for all versions of the XLog file. The times for the ten versions of each type of the XLog file were compared. This experiment studies the effects of changing the type and the number of processes in the performance of creating an XLog file.

9.6.2.2 Reading the XLog File

This Section explains the second part of the logging experiment that handles retrieving data from the XLog file. The reading process includes scanning the XLog file, searching for bad transactions and errors for a specific user and counting their number. The technical setup for the reading process is that the count of bad transactions and errors are calculated; the required time for reading the XLog file is measured. This step was repeated with thirty versions of the XLog file. The results for time consumed were compared from the perspectives of both the processes' type and number. Thus, this part of experiment evaluates retrieving data performance for all XLog files and finds the effects of changing the type and the number of processes on the time required to read the XLog file.

9.6.3 The Access Control Module Experiment

This experiment tests the access process from the functionality, scalability, and performance perspectives. It evaluates the access time for the access control module with different databases and numbers of users. The technical points of the experiment setup are described here. The access request is received as an XML query. Then, the Trust Value for the required node in the XML database's permission policy file is found and the time consumed is measured in milliseconds. The search process runs again to find the Trust Value

for the user in the users' access permission policy file and the required time is measured in milliseconds. After finding the Trust Value for the subject and the object, a comparison is made. If the TV for user is equal or larger than the TV for the node, the access decision allows the user to access the data. Otherwise access is denied. When the access is permitted, the time consumed for retrieving the data from XML databases, which is affected by the size of database, is measured in milliseconds. Finally, the total time for the access control module is measured in milliseconds.

This experiment was performed with different data sets (SIGMOD Record, NASA, and Treebank) and with different sizes of the user's access permission policy files (50, 100, and 1,000 users). The results include the time consumed in searching for the Trust Value of the user, finding the Trust Value of the node, the retrieval time, and the total time. The experiment compares the access control performance from two perspectives: the size of XML database and the number of users in the organisation

9.6.4 The Trust Maintenance Experiment

This experiment evaluates the performance of the whole system, integration between the trust module and the access control module. It measures the time required to perform the trust maintenance process. As mentioned in Chapter 6, trust maintenance includes three sub processes: evaluating for bad transactions and errors recorded in the XLog file, the calculation of Trust Value, and updating user privileges.

The technical points of the experiment setup are described here. The setup starts by reading the XLog file and counting the number of bad transactions (BTNum) and errors (ENum). Then, these numbers are evaluated and assigned the equivalent value for the Bad Transaction Factor (BTF) and Error Factor (EF). The Trust Value (TV) is calculated depending on the existing Trust Value (ETV), the Bad Transaction Factor (BTF), the Error Factor (EF),

and their weights. After the calculation, the user Trust Value in the users' access permission policy file is updated. The time consumed to complete the whole process is measured in milliseconds.

These technical procedures were run twelve times with different number of bad transactions and errors and with three users' sets: 50, 100, and 1,000. It compares the total time and evaluates the performance of the trust maintenance process depending on the changes in the number of bad transactions and errors and with a different number of users.

9.6.5 The Access Supervision Experiment

Like the trust maintenance, the access supervision experiment tests both modules working together. The access supervision process includes three processes: the access process, detecting bad transactions and errors, and recording in the XLog file. This experiment tests the performance and the scalability of the access supervision process. It compares the time required to perform this process with three data sets (Treebank, NASA, and SIGMOD Record) and with three user sets (50, 100, and 1,000) in two situations, when the access is permitted and when it is denied.

Two simple queries are used in this experiment. The first query includes deleting the root node, which is classified as a bad transaction process. [Q1: Deleting /RootNode]. This query tests the performance of the access supervision when the access is denied. The second query is a normal query that includes searching for a specific node: [Q2: Retrieving a specific node //node1]. This simple query is used to evaluate the access supervision process when the access is permitted.

The experimental setup includes many steps. It starts by performing the access process that is described in the access control module experiment (7.6.3) with respect to whether the XML query includes bad transactions or errors. Detecting bad transactions and errors in the XML query depending on the

operation policy file and error policy file was discussed in Chapter 7. After that, the XLog file is created and the bad transaction detected is recorded. Finally, the time consumed to perform the access supervision process is measured in milliseconds. The experiment compares the total time depending on the size of XML databases and the changing in the number of users.

9.6.6 The Experiment to Determine The Cost of Trust Based Access Control (TBAC)

This experiment tests the whole system of Trust Based Access Control for XML database (TBAC). It also compares the system performance and scalability with and without Trust Based Access Control (TBAC) to find the real time cost of applying TBAC.

XMark benchmark was used to generate different sized XML databases. Eleven XML databases from XFile1 to Xfile11 were used. These databases were defined in Section 9.4.3. The three versions of the users' access permission policy file were used in this experiment, which include the sets with 50, 100, and 1000 users.

Two queries were used to test the read privilege in both systems. The first is a simple query [Q3: //site/open_auctions/open_auction/initial]. The second is a complex query that includes joins of ancestor-descendant [Q4: //listitem//keyword]. The technical points of the experiment setup include performing both queries with different data sets and user sets in the Trust Based Access Control for XML databases and the access system without TBAC. The time required for both systems is measured in milliseconds and compared to examine performance.

9.6.7 The Comparison with MAC Experiment

This experiment compares Trust Based Access Control for XML database (TBAC) with the traditional approach of the Mandatory Access Control for XML databases (MAC) designed by Zhu (2009).

The experimental setup is almost the same as the one described in the previous Section (9.6.6) except that six XML files were used rather than eleven because the Mandatory Access Control approach designed by Zhu (2009) had published results with two queries for only these six files sizes.

The experimental setup includes performing both queries (Q3 and Q4) in the Trust Based Access Control for XML databases with different databases and user sets. The time required for the Trust Based Access Control system is measured in milliseconds. The performance speed of the Trust Based Access Control (TBAC) is compared with the published results of the Mandatory Access Control system (MAC) (Zhu et al., 2007; Zhu et al., 2009).

9.7 Conclusion

To test and evaluate the Trust Based Access Control for XML databases, seven experiments were performed. This Chapter described the design of experiments and explained their objectives. It reviewed existing natural data sets and XML benchmarks and the selection between them for the experiments. Three real data sets are used in the five experiments and the XMark benchmark used in the comparative experiments. The user sets are defined to cover the number of employees in small, medium, and large organisations. The setup for each experiment was discussed including: inputs, outputs, data sets, user sets, procedures, and measurements. The following Chapter presents the results from the experiments.

10 RESULTS AND ANALYSIS

10.1 Introduction

In Chapter 9, the design of seven experiments to evaluate Trust Based Access Control for XML databases was described. They evaluated the system in terms of functionality, performance, speed, scalability, and security. The first experiment focused on the calculation of Trust Value (TV). The second experiment concerned evaluating the logging process. The third evaluated the access control module and the fourth and fifth experiments were designed to test the system's processes, which are the trust maintenance and the access supervision. Finally, the sixth and seven experiments compared the results of the Trust Based Access Control system with the normal access, without TBAC, and the Mandatory Access Control method.

This Chapter presents and analyses the results of all these experiments. The results of each experiment are discussed in individual Sections. The discussion in Sections 10.2 to 10.8 summarises the experimental design from Chapter 9 and shows the results in tabular and graphical form.

10.2 The Trust Module Experiment

This Section discusses the results of the trust module experiment designed to evaluate calculations of the Trust Value (TV).

10.2.1 The Experimental Design Summary

Recall from Chapter 9, the trust module experiment was designed to identify appropriate values for the various weights, calculate the Trust Value (TV), and evaluate its performance and flexibility. The Trust Value depends on

six factors: Existing Trust Value (ETV), Bad Transaction Factor (BTF), Error Factor (EF), Existing Trust Value Weight (ETVW), Bad Transaction Factor Weight (BTFW), and Error Factor Weight (EFW). The calculations depend on four equations that were described in detail in Chapter 7.

$$TV = \begin{cases} ETV * ETVW + (1 - BTF) * BTFW + (1 - EF) * EFW, & BTF = 0 \text{ and } EF = 0 \quad (1) \\ ETV * ETVW - BTF * BTFW, & BTF > 0 \text{ and } EF = 0 \quad (2) \\ ETV * ETVW - EF * EFW, & BTF = 0 \text{ and } EF > 0 \quad (3) \\ ETV * ETVW - BTF * BTFW - EF * EFW, & BTF > 0 \text{ and } EF > 0 \quad (4) \end{cases}$$

The Trust Value (TV) performance was tested varying these six factors. The relationship between Trust Value (TV) and each factor is described in Section 10.2.3. This experiment was run to find the results of Trust Value (TV) when there are bad transactions, errors, or both. At the same time, the calculations of Trust Value (TV) were evaluated from the flexibility perspective depending on the weights.

10.2.2 Analytic Procedures

This Section explains procedures used to obtain the results (see Section 8.2.3) for the trust module experiment. Each factor used in calculating a TV has many values. The statistical results for all factors are represented in tabular format to explain the overall calculations. In addition, graphical representations are used to show the relationship between TV and the various factors. The results will be analysed in the next Section (10.2.3) from two viewpoints: the existence of bad transactions and errors and the altered weights.

10.2.3 Results Analysis

This Section discusses the results of the first experiment. All case studies are tested using three starting points for the Existing Trust Value (ETV): 0.75, 0.50, and 0.25. The experiment includes those where there are errors but no bad transactions, those where there are bad transactions but no errors, and those where there are both

Table 10.1 shows various TV when there is an Error Factor without a Bad Transaction Factor. The weight for both errors and bad transactions are at the maximum 10%. Since there is no bad transaction, its weight is not considered in the calculations. While the Error Factor increases gradually, the TV also decreases steadily. If ETV=0.75 when the Error Factor is 0.25 then the TV drops to 0.575. The TV falls to 0.5 while the Error Factor reaches the maximum of 1. Figure 10.1 depends on data in Table 10.1 and represents graphically the relationship between the changes in Trust Value (TV) and various values of errors.

Table 10.1 The results with various values of the Error Factor without the Bad Transaction Factor

ETV	ETVW	EF	EFW	BTF	BTFW	TV
0.75	%80	0.25	%10	0	%10	0.575
0.50	%80	0.25	%10	0	%10	0.375
0.25	%80	0.25	%10	0	%10	0.175
0.75	%80	0.5	%10	0	%10	0.55
0.5	%80	0.5	%10	0	%10	0.35
0.25	%80	0.5	%10	0	%10	0.15
0.75	%80	0.75	%10	0	%10	0.525
0.5	%80	0.75	%10	0	%10	0.325
0.25	%80	0.75	%10	0	%10	0.125
0.75	%80	1	%10	0	%10	0.5
0.5	%80	1	%10	0	%10	0.3
0.25	%80	1	%10	0	%10	0.1

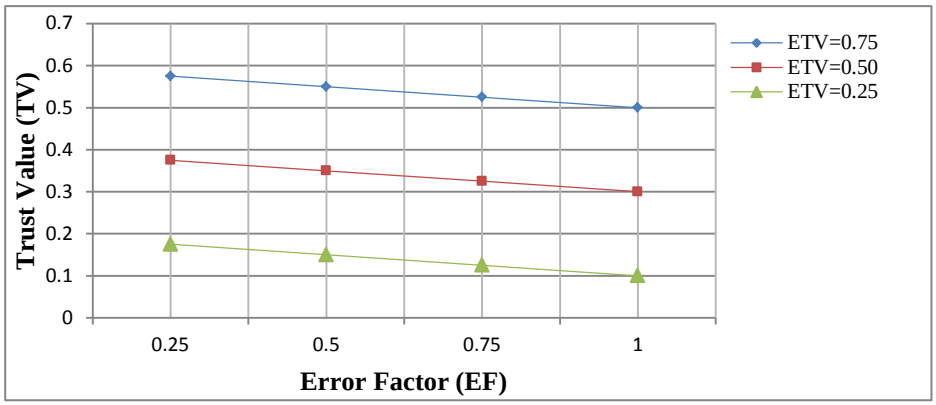


Figure 10.1 The relation between TV and EF

Table 10.2 illustrates the calculations of TV when there is only Bad Transaction Factor without the Error Factor. The results for TV are the same as in Table 10.1 because the same weights for errors and bad transaction are used. The Bad Transaction Factor again affects inversely in TV. While the Bad Transaction Factor increases, the TV decreases. This inverse relationship is displayed graphically in Figure 10.2.

Table 10.2 The results with only the Bad Transaction Factor (without the Error Factor)

ETV	ETVW	EF	EFW	BTF	BTFW	TV
0.75	%80	0	%10	0.25	%10	0.575
0.50	%80	0	%10	0.25	%10	0.375
0.25	%80	0	%10	0.25	%10	0.175
0.75	%80	0	%10	0.5	%10	0.55
0.5	%80	0	%10	0.5	%10	0.35
0.25	%80	0	%10	0.5	%10	0.15
0.75	%80	0	%10	0.75	%10	0.525
0.5	%80	0	%10	0.75	%10	0.325
0.25	%80	0	%10	0.75	%10	0.125
0.75	%80	0	%10	1	%10	0.5
0.5	%80	0	%10	1	%10	0.3
0.25	%80	0	%10	1	%10	0.1

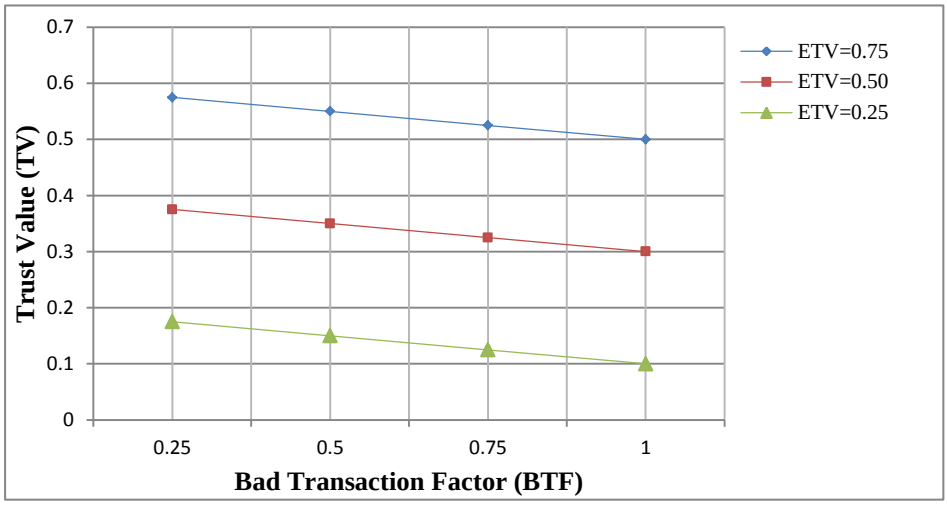


Figure 10.2 The relation between TV and BTF

After explaining the results with Error Factor only and bad transaction only, the results of TV when there is a mixture of errors and bad transactions are described in the following diagrams. The results are measured with four categories of weights: the maximum, the minimum, the intermediate, and the recommended weights. The effects of changes in error and bad transaction weights are now discussed. As mentioned in Section 7.6, both Bad Transaction Factor Weight (BTFW) and Error Factor Weight (EFW) range between 1% and 10%. The Existing Trust Value Weight (ETVW) range is between 80% and 98%.

Demonstrations of the variation in Trust Value with different values for both errors and bad transactions are summarised in Table 10.3, Table 10.4, Table 10.5, and Table 10.6. Table 10.3 explains the statistical results when the error and bad transaction weights are at the maximum value and the Existing Trust Value Weight is at its minimum: ETVW=80%, EFW=10%, and BTFW=10%. The results show how the Trust Value is changed significantly due to the effects of these weights. Using data in Table 10.3, Figure 10.3

graphically displays the relation between TV and the mixture of errors and bad transactions with the maximum weights.

Table 10.3 The results when errors and bad transactions' weights are at the maximum

ETV	ETVW	EF	EFW	BTF	BTFW	TV
0.75	%80	0.25	%10	0.25	%10	0.55
0.50	%80	0.25	%10	0.25	%10	0.35
0.25	%80	0.25	%10	0.25	%10	0.15
0.75	%80	0.5	%10	0.5	%10	0.5
0.5	%80	0.5	%10	0.5	%10	0.3
0.25	%80	0.5	%10	0.5	%10	0.1
0.75	%80	0.75	%10	0.75	%10	0.45
0.5	%80	0.75	%10	0.75	%10	0.25
0.25	%80	0.75	%10	0.75	%10	0.05
0.75	%80	1	%10	1	%10	0.4
0.5	%80	1	%10	1	%10	0.2
0.25	%80	1	%10	1	%10	0

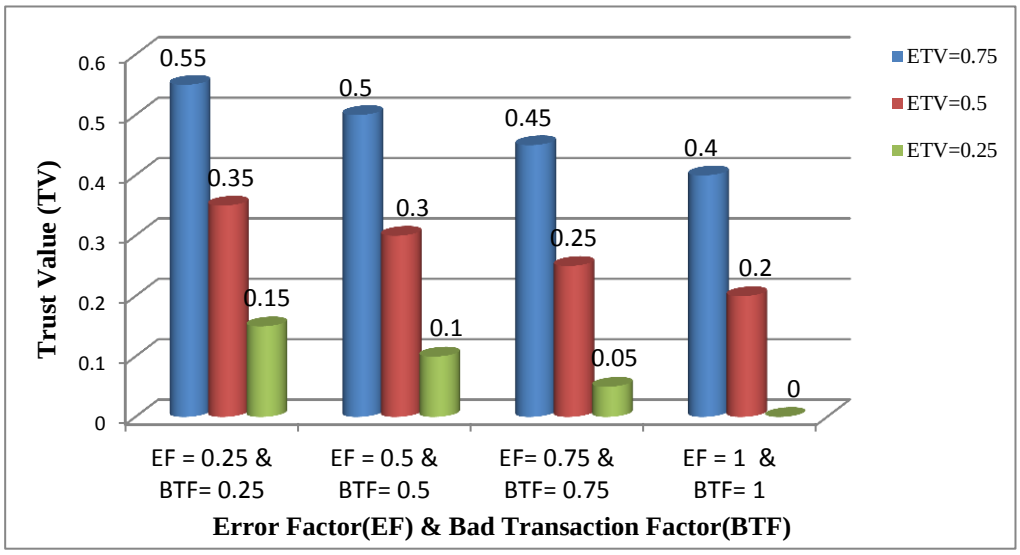


Figure 10.3 The comparative results of various values for both errors and bad transactions with the maximum weights

Table 10.4 shows the results when the error and bad transactions' weights are at the minimum and the Existing Trust Value Weight is at its maximum; therefore, ETVW=98%, EFW=1%, and BTFW=1%. Using these weights means the new Trust Value is only slightly affected by errors and bad transactions. This reflects that the organisation considers that both the Error Factor and the Bad Transaction Factor are not important in the user evaluation process. The outcomes show that, using these weights, the TV is relatively static. Figure 10.4 represents graphically the relation between TV and the mixture of errors and bad transactions with the minimum weights.

Table 10.4 The results when errors and bad transactions' weights are at the minimum

ETV	ETVW	EF	EFW	BTF	BTFW	TV
0.75	%98	0.25	%1	0.25	%1	0.73
0.50	%98	0.25	%1	0.25	%1	0.485
0.25	%98	0.25	%1	0.25	%1	0.24
0.75	%98	0.5	%1	0.5	%1	0.725
0.5	%98	0.5	%1	0.5	%1	0.48
0.25	%98	0.5	%1	0.5	%1	0.235
0.75	%98	0.75	%1	0.75	%1	0.72
0.5	%98	0.75	%1	0.75	%1	0.475
0.25	%98	0.75	%1	0.75	%1	0.229
0.75	%98	1	%1	1	%1	0.715
0.5	%98	1	%1	1	%1	0.47
0.25	%98	1	%1	1	%1	0.224

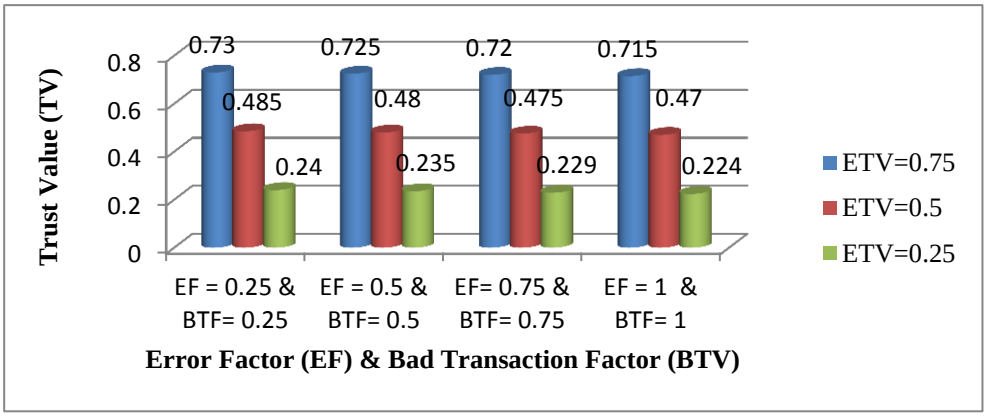


Figure 10.4 The comparative results of various values for both errors and bad transactions with the minimum weights

These calculations were then repeated with the intermediate weight. Table 10.5 shows the calculations results with the intermediate weight for both the Error Factor and Bad Transaction Factor. The results illustrate that the weights at 5% reasonably affect the calculations.

Table 10.5 The results when errors and bad transactions' weights are at the middle range

ETV	ETVW	EF	EFW	BTF	BTFW	TV
0.75	%90	0.25	%5	0.25	%5	0.65
0.50	%90	0.25	%5	0.25	%5	0.425
0.25	%90	0.25	%5	0.25	%5	0.199
0.75	%90	0.5	%5	0.5	%5	0.625
0.5	%90	0.5	%5	0.5	%5	0.399
0.25	%90	0.5	%5	0.5	%5	0.175
0.75	%90	0.75	%5	0.75	%5	0.6
0.5	%90	0.75	%5	0.75	%5	0.375
0.25	%90	0.75	%5	0.75	%5	0.15
0.75	%90	1	%5	1	%5	0.575
0.5	%90	1	%5	1	%5	0.350
0.25	%90	1	%5	1	%5	0.124

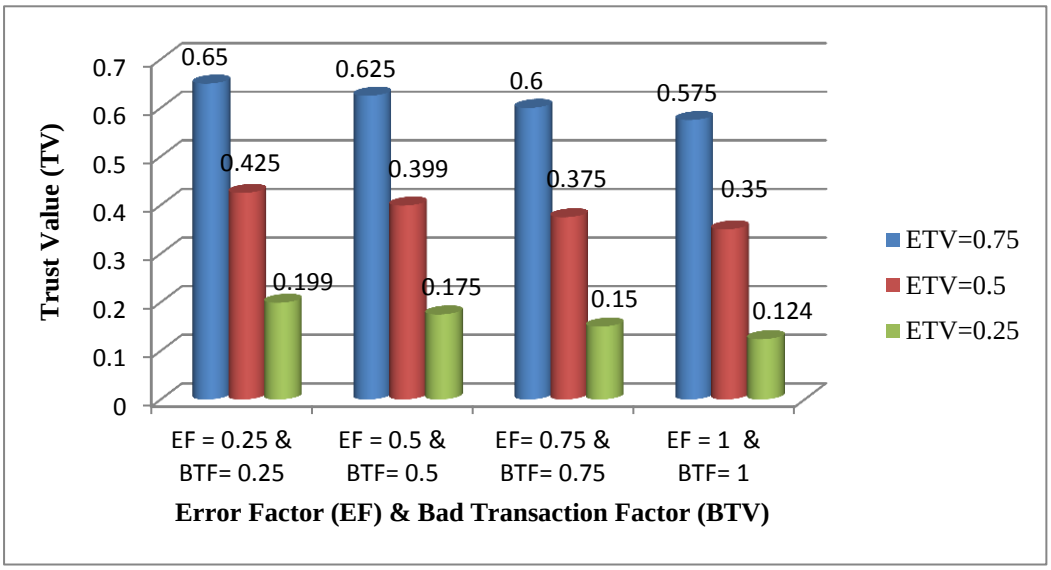


Figure 10.5 The comparative results of various values for both errors and bad transactions with the middle range weights

The weights are flexible and can be changed with the organisation's policy. Compared to the results in Table 10.3, Table 10.4, and Table 10.5, the recommended weights are ETVW=85%, EFW=5%, and BTFW=10%. The Error Factor Weight is selected to be 5%, the middle range value, because an error cannot be as harmful as a bad transaction. For the Bad Transaction Factor Weight, the weight is recommended to be the highest value allowed, which is 10%, because a bad transaction could reflect malicious intent. The results of these recommended weights are shown in Table 10.6. It appears that the Trust Value changes regularly and markedly for all starting points of the Existing Trust Value. The relation between TV and the error and Bad Transaction Factors is displayed in Figure 10.6.

Table 10.6 The results when errors and bad transactions' weights are at the recommended weights

ETV	ETVW	EF	EFW	BTF	BTFW	TV
0.75	%85	0.25	%5	0.25	%10	0.6
0.50	%85	0.25	%5	0.25	%10	0.387
0.25	%85	0.25	%5	0.25	%10	0.175
0.75	%85	0.5	%5	0.5	%10	0.562
0.5	%85	0.5	%5	0.5	%10	0.35
0.25	%85	0.5	%5	0.5	%10	0.137
0.75	%85	0.75	%5	0.75	%10	0.524
0.5	%85	0.75	%5	0.75	%10	0.312
0.25	%85	0.75	%5	0.75	%10	0.099
0.75	%85	1	%5	1	%10	0.487
0.5	%85	1	%5	1	%10	0.275
0.25	%85	1	%5	1	%10	0.062

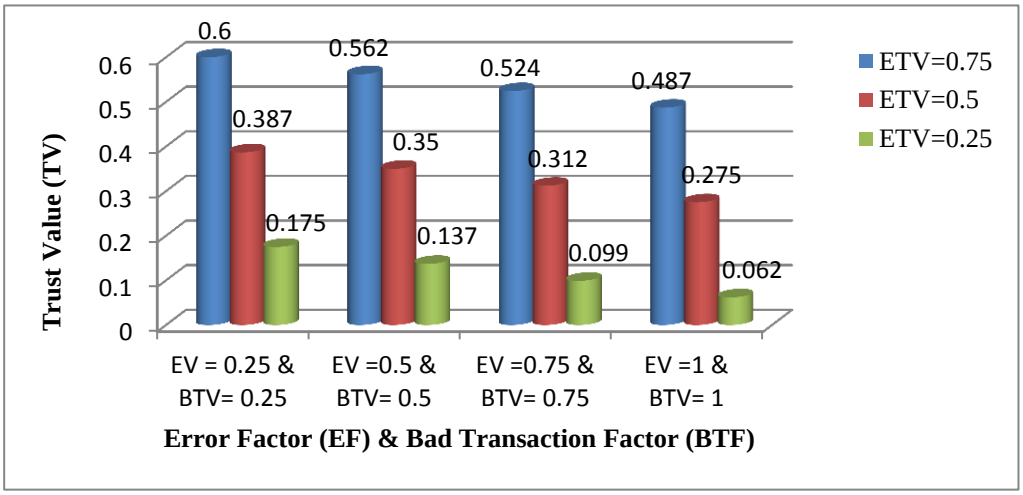


Figure 10.6 The comparative results of various values for both errors and bad transactions with the recommended weights

Having demonstrated the decrement in TV depending on various values of the Error Factor and Bad Transaction Factor, the increment in TV is now discussed. Table 10.7 displays the increment in Trust Value (TV) when there is

no error and bad transaction whatsoever. The outcomes show different case studies. The calculations are made with the minimum, the maximum, the intermediate, and the recommended weights. Figure 10.7 compares the increment in TV with different weights.

Table 10.7 The results when there are no errors or bad transactions whatsoever

ETV	ETVW	EF	EFW	BTF	BTFW	TV
0.75	%80	0	%10	0	%10	0.8
0.50	%80	0	%10	0	%10	0.6
0.25	%80	0	%10	0	%10	0.4
0.75	%85	0	%5	0	%10	0.787
0.50	%85	0	%5	0	%10	0.575
0.25	%85	0	%5	0	%10	0.362
0.75	%90	0	%5	0	%5	0.775
0.50	%90	0	%5	0	%5	0.55
0.25	%90	0	%5	0	%5	0.325
0.75	%98	0	%1	0	%1	0.755
0.50	%98	0	%1	0	%1	0.51
0.25	%98	0	%1	0	%1	0.265

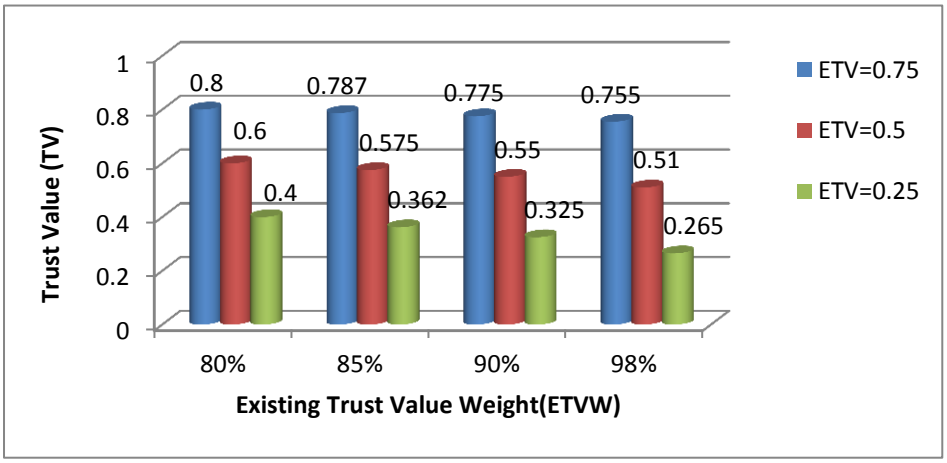


Figure 10.7 The comparative results for different values of ETVW with no errors or bad transactions

10.2.4 Conclusion

The experimental results described the changes in TV depending on the values of errors and bad transactions and explained the effects of weights. After analysing the results, the recommendations for weights were made. Both increment and decrement in the calculations was displayed in graphical forms. In general, the results of this experiment were as expected. The TV increased when there were no errors and bad transactions whatsoever. It decreased if there were errors, bad transactions, or both. This experiment showed the flexibility of calculating TV for users, which supports using dynamic Trust Based Access Control to amend users' privileges based on their operations to prevent insider threats.

10.3 The Logging Experiment

This Section shows and analyses the results for the logging experiment designed to evaluate logging in Trust Based Access Control for XML databases.

10.3.1 The Experimental Design Summary

This experiment was divided into two parts. The first part handled the creation process of the XLog file. The second focused on the reading from the log file. The main goal of the creation experiment is to measure the time required to create the XLog file and record processes. The retrieval experiments were used to evaluate the performance of reading the XLog files.

10.3.2 Analytic Procedures

Both the creation and retrieval experiments were executed with three types of XLog files. The first file type consists of errors. The second file type includes only bad transactions. The third has errors with bad transactions. These three XLog files were then tested in several steps depending on the number of recorded processes. The first and second XLog files start from 10 processes with

either errors or bad transaction until they reach 100 processes. In each step, the number of processes is increased by 10. Since the third XLog file has both errors and bad transactions, the number of processes is shared equally between them. This third XLog file runs from 20 to 200 processes. The number of processes is increased by 10 errors and 10 bad transactions each step.

These experiments were executed by using two test factors, which are the process types and the number of processes. In the next Section, the results are described in graphical representations and the comparison between two processes is made.

10.3.3 Results Analysis

10.3.3.1 The Creation Process for the XLog File

The results of creating the first XLog file that contains errors only and the second that includes only bad transactions were almost identical. This is because the time consumed for recording the same number of processes in different XLog files is similar even if the type of process is different. As result the creation of the XLog file is affected by the number of recorded processes regardless of their type. The time required to create the XLog file with 10 processes with errors or bad transactions is 122 milliseconds. This time increases steadily by around 50 milliseconds when the number of errors or bad transactions is raised by 10 processes each time. When the number of either errors or bad transactions is 100 processes, the time reaches around 566 milliseconds. Figure 10.8 illustrates the outcomes of the creation of the XLog file with only errors or only bad transactions.

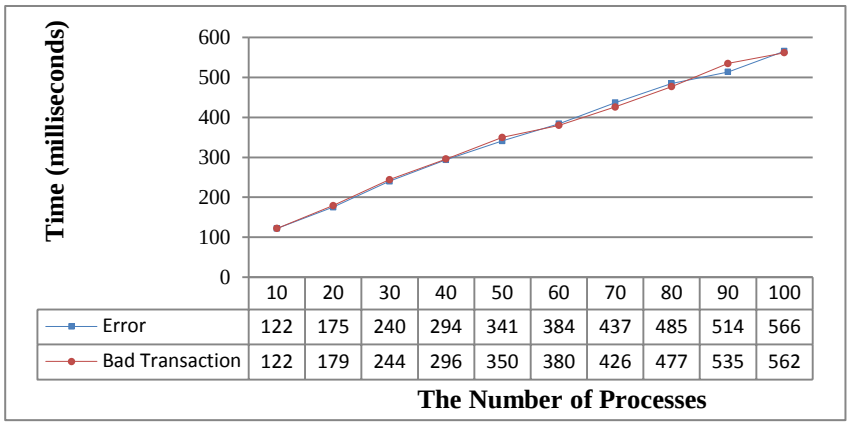


Figure 10.8 The comparatives results of creating the XLog file with errors only or bad transaction only

The time taken to create the third XLog file, which has a mixture of errors and bad transactions, is increased by around 100 milliseconds when the number of processes is increased by 20 processes. At the start, when the number of errors is 10 and the number of bad transactions is 10, the time for creating the XLog file with these 20 processes is 186 milliseconds. The creation time grows markedly until reaching 1,098 milliseconds when there are 200 processes, which are 100 errors and 100 bad transactions. Figure 10.9 shows the result of creating the XLog file with both errors and bad transactions.

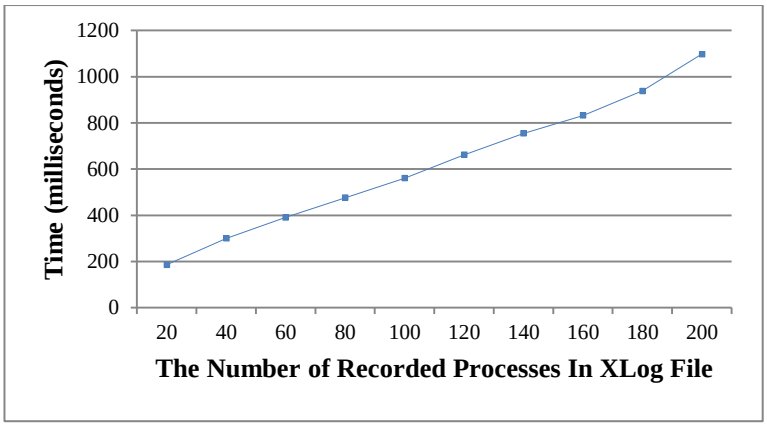


Figure 10.9 The required time for creating the XLog file with both errors and bad transactions

10.3.3.2 Reading the XLog File

The time required for reading the XLog files that contained either only errors or only bad transactions is again similar. The time starts to increase from 79 milliseconds to read 10 processes up to 103 milliseconds to read 100 processes. The time is increased by around four milliseconds when the number of errors or bad transaction is increased by 10 processes. Figure 10.10 represents the time required to read an XLog file that contains only errors or only bad transactions.

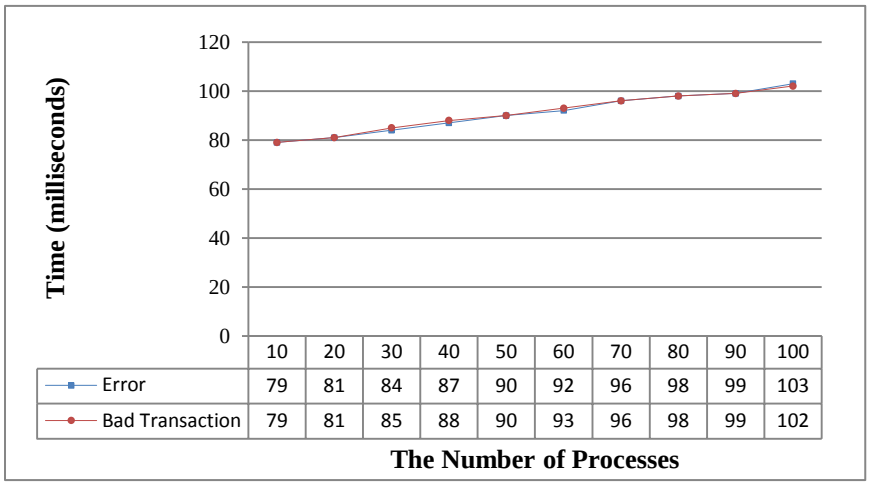


Figure 10.10 The comparatives results of reading the XLog file with errors only or bad transaction only

The time consumed to read the third XLog file type, which has both errors and bad transactions, increases by around six milliseconds when the number of processes grows by 20 processes. The time to access the third XLog file starts from 82 when the number of processes is 20 and ends with 125 milliseconds when the number of processes is 200. Figure 10.11 explains the results of reading and accessing the XLog file when half of the processes are errors and the others are bad transactions.

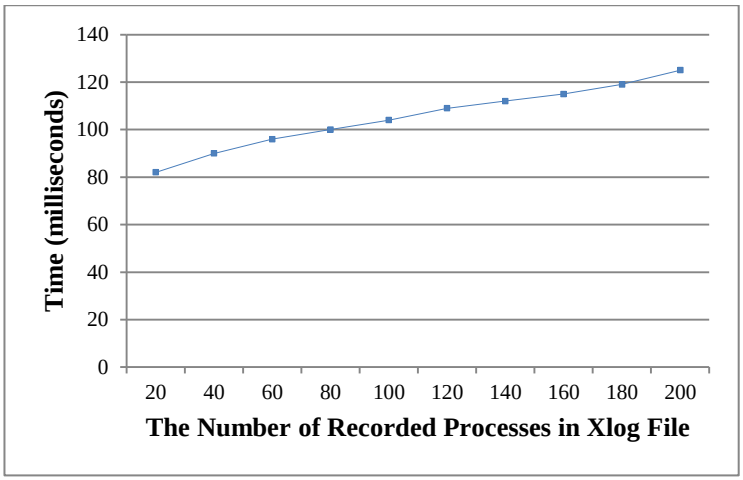


Figure 10.11 The required time for reading the XLog file with both errors and bad transactions

The results of the creation and reading processes with both errors and bad transactions are compared in Figure 10.12. The comparison shows that the reading process is faster than the creation process for the XLog file. The time required for the creation increases significantly depending on the number of processes. In contrast, the time required for reading increases steadily while the number of processes grows.

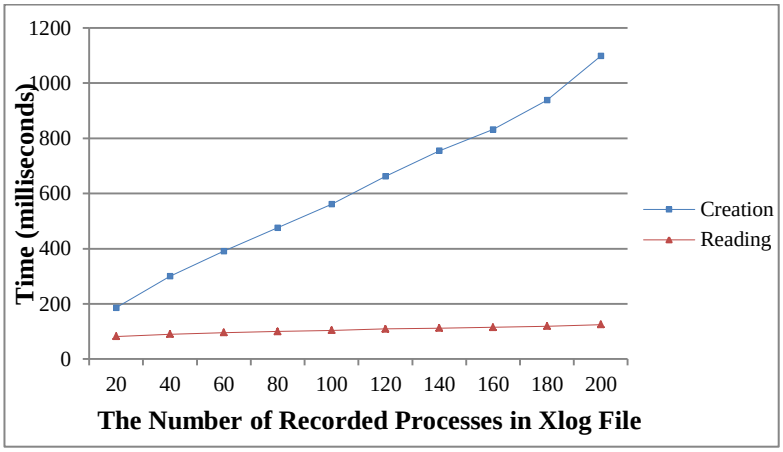


Figure 10.12 The comparative results of creation and reading processes with both errors and bad transactions

10.3.4 Conclusion

This experiment focused on evaluating the creation of the Xlog file and reading it. This evaluation ensures that the Xlog file worked properly, which improves the security level and supports Trust Based Access Control for XML databases. In general, the time consumed for the reading process was always less than the time for creation. Both processes consumed a reasonable time: this was expected based on the dynamic design of the XLog file. The creation and retrieval processes were not affected by the type of processes, whether errors or bad transactions. Both experiments are affected markedly by changing the number of processes in each step.

10.4 The Access Control Module Experiment

In this Section, the results of the access control module experiment to evaluate the performance of the access module in Trust Based Access Control for XML databases are discussed.

10.4.1 The Experimental Design Summary

As mentioned in Chapter 9, this experiment tested the access process in the access control module from performance, speed and scalability perspectives. To evaluate the access process performance, the access time taken with different databases and different numbers of users were measured. The access time consists of the search time for both the user and data Trust Values and the retrieval time from the original databases. The accessibility evaluation depends on accessing processing time for the three data sets with the three different numbers of users. The experimental results are compared in Section 10.4.3. Full results detail are shown in APPENDIX I.

10.4.2 Analytic Procedures

The access processing evaluation is executed in three steps:

- Checking user's TV search time: this is the time to find the user Trust Value in the user access permission policy file.
- Checking data TV search time: the time to find the data's Trust Value in the XML database permission policy file.
- Checking node searched time: this is the time to retrieve the node information from the original XML database.

Three real XML databases with different file sizes were tested: SIGMOD Record, NASA, and Treebank. This experiment was performed with a number of user access permission policy files with different number of users: 50, 100, and 1,000. The results of each step are compared. The total required time for the entire access process is described. All results are displayed graphically in the next Section.

10.4.3 Results Analysis

The results include the search time for the user, the node and data, and the total time. Figure 10.13 shows the user's TV search time for different XML databases with different number of users. In general, the time taken to find the user Trust Value increases directly with the growth in the number of users. It seems that the data set's size does not cause marked changes in user TV search time. When the number of user is 50, the time is around 4 milliseconds. When the number of users is 100, the time is around 7.5 milliseconds. The time rises to reach 20 milliseconds when the number of user is 1,000.

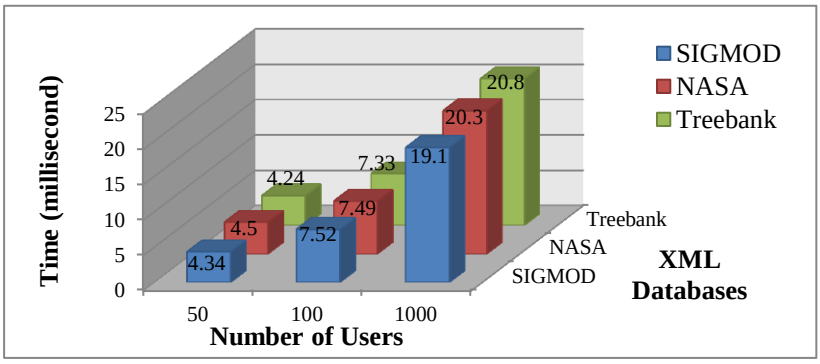


Figure 10.13 User TV search time

Figure 10.14 shows the result of searching data Trust Value in the XML database permission policy file with different data sets and different number of users. Since the XML database permission policy file is small and only contains the nodes' structure without the repetition in the original XML database, the time consumed to find the data Trust Value is short. The time is not affected by the number of users. Correspondingly, it does not depend on data sets' size and is only slightly affected by the XML database structure. The time for SIGMOD is around 1.4 milliseconds, NASA is about 2 milliseconds, and Treebank is 1.5 milliseconds.

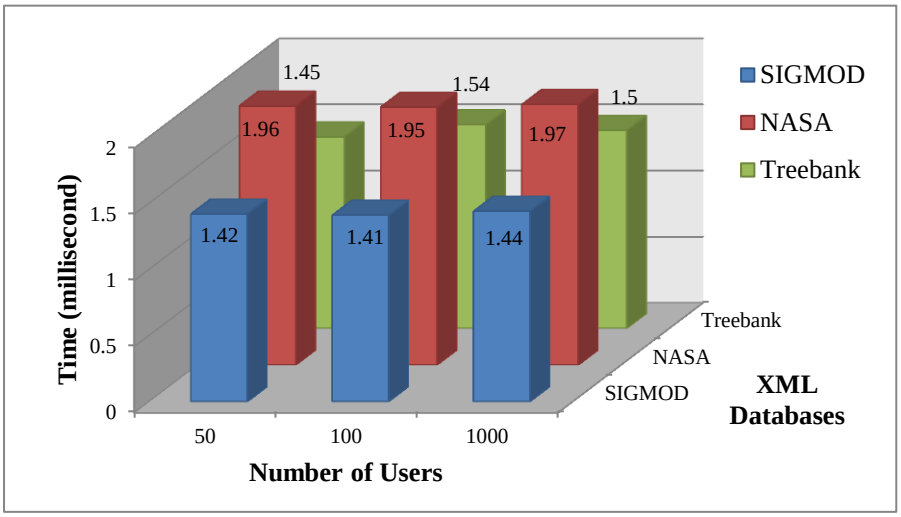


Figure 10.14 Data TV search time

Figure 10.15 presents the result of retrieving data from the XML databases when access is permitted. It is evident that the time is affected mostly by XML databases' sizes and it is not significantly affected by the changes in the number of users. The time for SIGMOD is nearly 37 milliseconds, NASA 724 milliseconds, and Treebank 1,990 milliseconds.

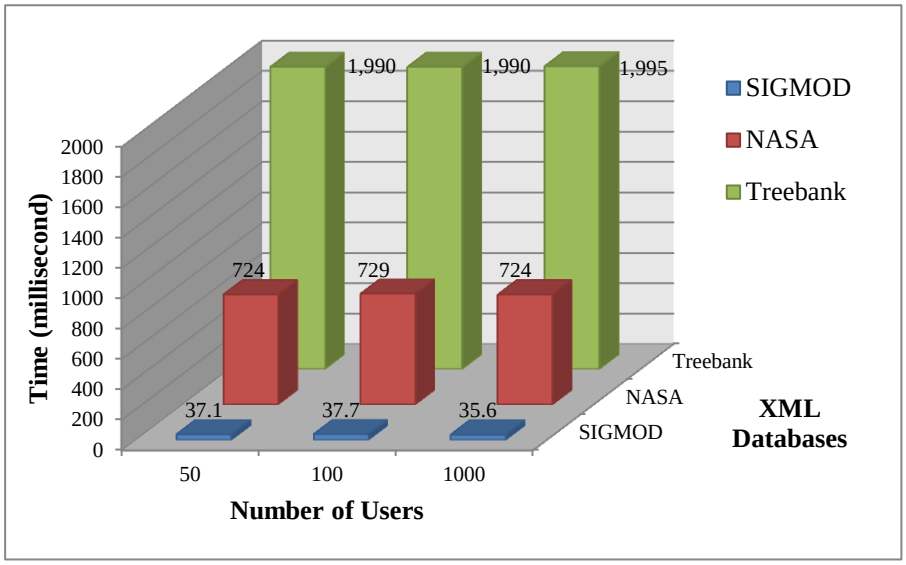


Figure 10.15 Node search time

Figure 10.16 shows the total access time for a specific node in three databases with different numbers of users. In general, this time is affected significantly by the retrieval time, which means that the final access time is dependent mainly on the size of the database. The time for SIGMOD is around 0.15 seconds, NASA is about 2 seconds, and Treebank is 5.6 seconds.

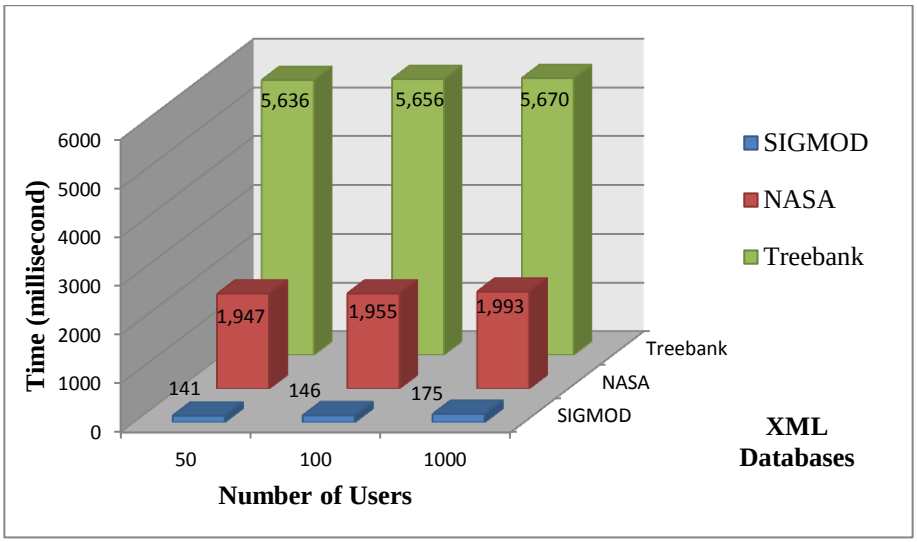


Figure 10.16 The access time using a variety of different sized users' sets

10.4.4 Conclusion

This experiment tested the performance and the scalability of the access control module to large XML databases and different numbers of users. Testing the access control module ensures that the access process is performed efficiently by checking users' privileges based on TV, which prevents unauthorised transactions. The results showed that the access module worked with small (SIGMOD), medium (NASA), and large (Treebank) XML databases. The system was tested with different number of users to cover small (50), medium (100), and large (1,000) organisations. The time needed to discover a user's Trust Value is affected by the number of users. The time to find data Trust Value is slightly affected by the XML databases' structure. The retrieval time is mainly affected by the size of the databases. The total access time is affected chiefly by the retrieval time. The access performance is better than expected. This is because searching processes for users' TV and data TV were relatively fast due to dividing policy rules into sub files.

10.5 The Trust Maintenance Experiment

Combining the trust module and the access control module generates the system processes. These processes are classified into trust maintenance and access supervision (see Chapter 6). This Section describes the results of the trust maintenance experiment.

10.5.1 The Experimental Design Summary

This experiment was designed to test the performance, speed, and scalability in the trust maintenance process. The maintenance process consists of three sub processes: evaluating errors and bad transactions, calculating the Trust Value (TV), and updating users' Trust Values. This process runs at a frequency depending on the organisation policy and is used to evaluate users' behaviour and update their privileges. It could be performed hourly, daily, weekly, or monthly.

10.5.2 Analytic Procedures

The experiment was performed to evaluate the time required to accomplish the whole process. The total time of the trust maintenance includes:

- The evaluation time: the required time to read the XLog file and count errors and bad transactions.
- The calculation time: this time is used to calculate the new TV using the equations, given in Section 7.6. The calculation process is described in Chapter 7 and its experiment was discussed in Section 10.2.
- The update time: the time needed to update the user Trust Value in the users' access permission policy file. Updating the TV for the user means changing the user's privilege to access the system.

This experiment was performed with three versions of the users' access permission policy file that include 50, 100, and 1,000 users. The results have

been compared graphically depending on the values of both the Error Factor and the Bad Transaction Factor.

10.5.3 Results Analysis

The time consumed for processing the trust maintenance is explained in Figure 10.17. The time was relatively short for all case studies. When the number of users was 50, the time was around 0.12 seconds for all values of the Error Factor and the Bad Transaction Factor. The time is 0.13 seconds with 100 users and 0.18 with a 1,000 user sample. Changes in both the Error Factor and the Bad Transactions Factor do not affect the total time. The time increases slowly when the number of users grows. Extra results of repeating this experiment in platform two are in APPENDIX II and are consistent with these results.

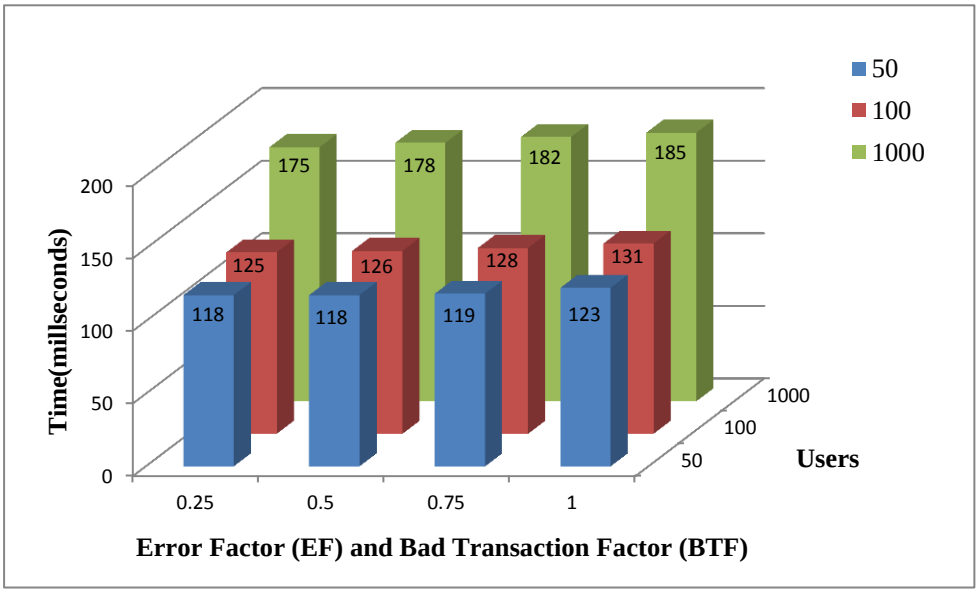


Figure 10.17 The time consumed for the trust maintenance process

10.5.4 Conclusion

This experiment aimed to evaluate the performance speed of the trust maintenance process. This evaluation confirms that security features worked properly in the system since this process included the evaluation errors and bad transactions, calculating TV, and updating privileges. The results show that running this process is quite fast and requires little time. The time is only slightly affected by the number of users in the system. As expected, it appears that the short and dynamic design of the XLog file leads to an efficient performance of the trust maintenance process.

10.6 The Access Supervision Experiment

The access supervision process is the second main process in the system. This Section discusses the experiment to test this process from the perspectives of functionality, performance, and scalability. The results are represented graphically in Section 10.6.3. Extra results for platform two are given in APPENDIX III.

10.6.1 The Experimental Design Summary

This experiment was designed to evaluate the access supervision process. The process includes three mini processes: access, detecting errors and bad transactions, and recording in the log file. It is performed for each access request. The experiment tested its speed and its performance in both cases; when the access is permitted and denied.

10.6.2 Analytic Procedures

The time required to perform the access supervision process is measured in milliseconds. This total time includes:

- The access time: when the access is permitted, this is the time required to access and retrieve data from the XML databases. This was described in detail in the access control module experiment (see Section 10.4).
- The detection time: the time needed to capture errors and bad transaction in the XML query.
- The recording time: the time to record errors and bad transactions in the XLog file.

The performance of the supervision process was tested through two queries. The first query consisted of deleting the root node from the XML databases [Q1: Deleting/RootNode]. This query is denied and is recorded as a bad transaction. The second query concerns retrieving a specific node from the XML databases [Q2: //NodeName]. This query is a permitted access. The total time for both situations is compared with three different data sets: SIGMOD Record, NASA, and Treebank. Three user sets were also used in the evaluation process.

10.6.3 Results Analysis

The result of the supervision process when access is denied and the operation is recorded in the XLog file as a bad transaction is illustrated in Figure 10.18. The time consumed to complete the supervision process for the first query is relatively short because the retrieval time from the original databases does not include any access. The time needed includes the search time for the Trust Values for both users and data and the time required for logging. In this case study, the time is slightly affected by the number of users. For three different data sets, the time consumed is around 87 milliseconds when the number of users is 50. The time is 96 milliseconds for 100 users and around 128 milliseconds for 1,000 users.

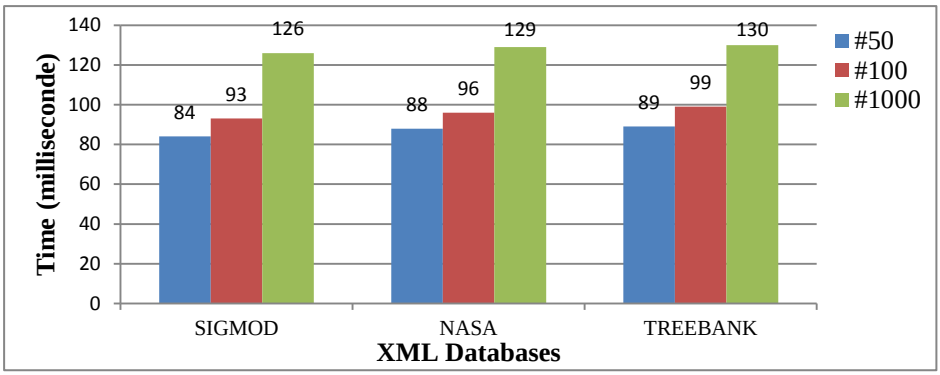


Figure 10.18 The time consumed for processing the access supervision process in the first query (Q1)

Figure 10.19 shows the results when the access is permitted using the second query (Q2). The supervision process takes longer to execute the second query. The final access decision for the second query is to permit it, so that the access time includes the retrieval time, which depends on the XML database size. The time increases with the growth in the size of XML databases. The time is between 155 and 183 milliseconds for SIGMOD Record. The time range for NASA is from 2,039 milliseconds with 50 users to 2,086 milliseconds with 1,000 users. The total time significantly increases in Treebank due to the size of the XML database. It reaches 6,101 milliseconds with 1,000 users.

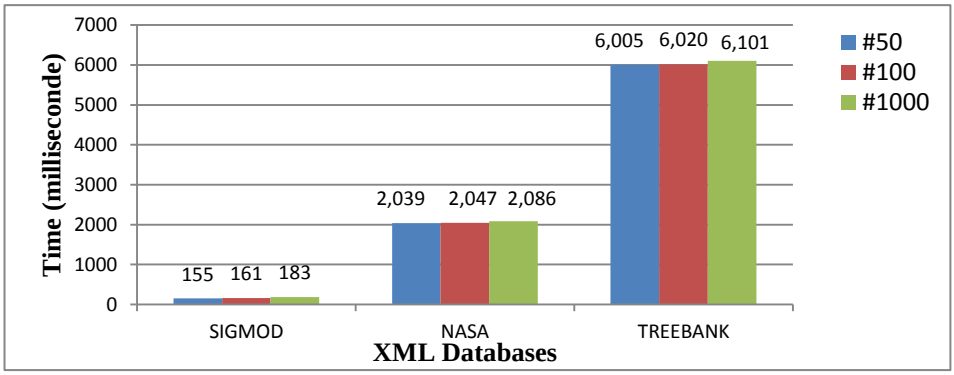


Figure 10.19 The time consumed for processing the access supervision process in the second query (Q2)

The comparative results of the time consumed for supervising the process in the system for both situations (Q1 and Q2) with 1,000 users are summarised in Figure 10.20.

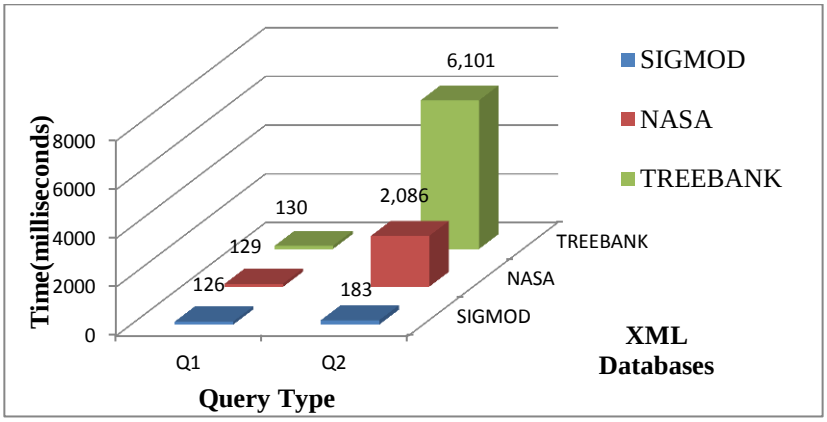


Figure 10.20 The time consumed for processing the access supervision process with 1,000 users

10.6.4 Conclusion

This experiment evaluated the speed of the access supervision process. Ensuring the performance of this process leads to implicitly improve the security aspect in the system since it consists of accessing, capturing errors and bad transactions, and recording them in the Xlog file. The results show, as expected, that the process runs faster when the access is denied than when the access is permitted. The time required when the access is allowed is affected by the XML databases' size since it includes the retrieval time. Although the access supervision includes three complex sub processes, its general performance is acceptable, which was unexpected. The experiment shows the scalability in the access supervision from experiments performed with three data sets and three users' sets.

10.7 The Experiment to Determine the Cost of Trust Based Access Control (TBAC)

In this Section, the use of Trust Based Access Control is evaluated by measuring the real time cost. The comparison experiment for the system with and without TBAC is described in Section 10.7.3.

10.7.1 The Experimental Design Summary

The main goal of this experiment is to evaluate the performance of Trust Based Access Control. This experiment aims to evaluate the time required to apply Trust Based Access Control for XML databases by comparing the time with and without this control.

10.7.2 Analytic Procedures

This experiment evaluated the access time with and without Trust Based Access Control. It was performed in two steps. The first step measured the real time cost of applying Trust Based Access Control for XML databases. The second compared the results with the normal access time.

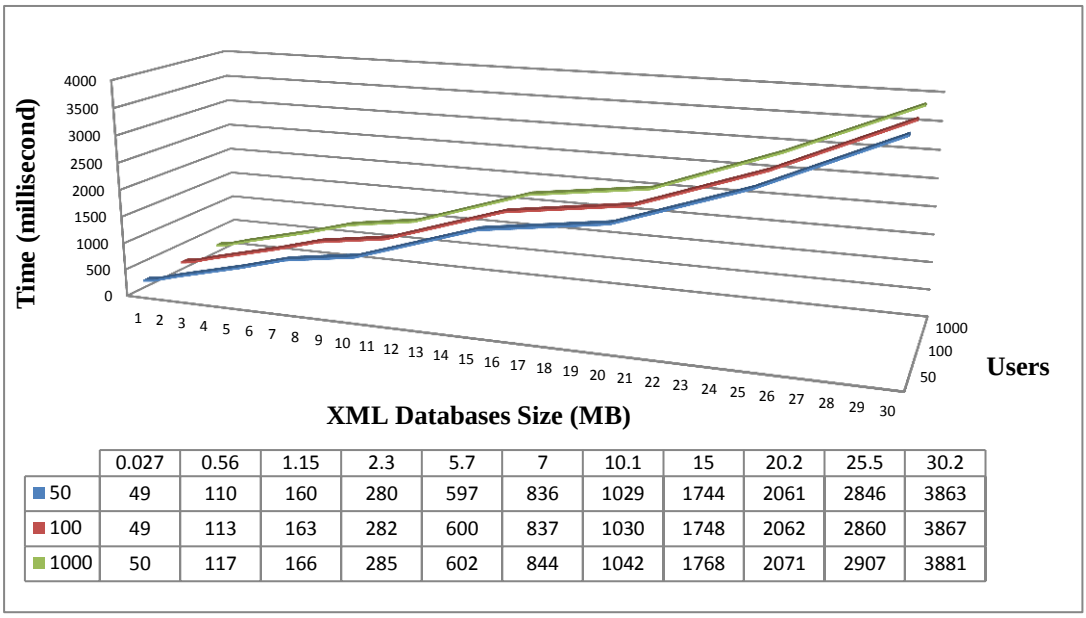
The first part of the experiment tested the scalability in XML databases and the number of users. XMark benchmark was used to generate eleven XML databases with different file sizes. Table 11.3 shows the size of each XML database used in the experiment. Three user sets (50, 100, and 1,000) were used in the first step of experiment.

For simplicity, only read privilege was tested through two queries. The first was a simple query [Q3: //site/open_auctions/open_auction/initial]. The second was a complex query that includes joins of ancestor-descendant [Q4: //listitem//keyword].

The normal access time was again tested through these queries. The comparative results are represented graphically in the next Section. In the comparison, the number of users was selected to be 50 as its minimum because the main aim of this experiment is to measure the access time with and without Trust Based Access Control and not to check the scalability in the number of users.

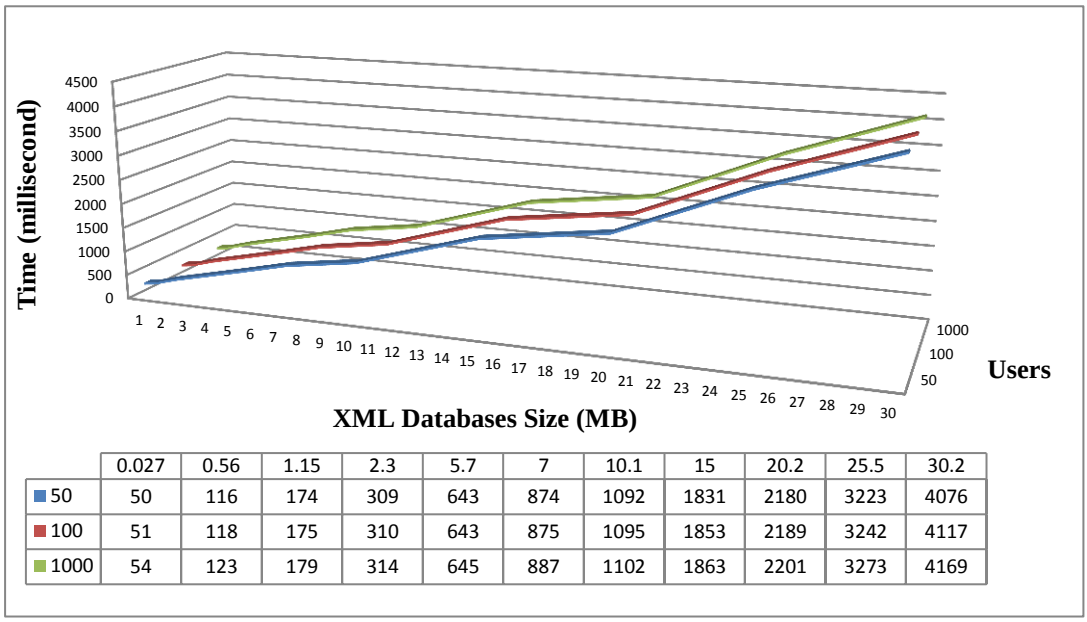
10.7.3 Results Analysis

The results of Trust Based Access Control for XML databases with the simple query are summarised in Figure 10.21 and those with the complex query are illustrated in Figure 10.22. The results for Trust Based Access Control are affected markedly by changing the size of the XML database. The real time consumed increases as the database size increases. The results with three user sets in both queries are very similar, which means that changing the number of users does not have great consequence in the system. The performance of the simple query is slightly faster than the complex query. The time for simple query starts with 50 milliseconds when the XML database size is 0.027MB and reaches 3.8 seconds when the XML database size is 30.2MB. In the results of the complex query, the time consumed is 54 milliseconds when the XML database size is 0.027MB and it is 4.1 seconds when the size is 30.2MB.



	0.027	0.56	1.15	2.3	5.7	7	10.1	15	20.2	25.5	30.2
50	49	110	160	280	597	836	1029	1744	2061	2846	3863
100	49	113	163	282	600	837	1030	1748	2062	2860	3867
1000	50	117	166	285	602	844	1042	1768	2071	2907	3881

Figure 10.21 The results of TBAC in the simple query



	0.027	0.56	1.15	2.3	5.7	7	10.1	15	20.2	25.5	30.2
50	50	116	174	309	643	874	1092	1831	2180	3223	4076
100	51	118	175	310	643	875	1095	1853	2189	3242	4117
1000	54	123	179	314	645	887	1102	1863	2201	3273	4169

Figure 10.22 The results of TBAC in the complex query

After finding the real time required for Trust Based Access Control, the comparative results with the normal access time are illustrated in the following figures. Figure 10.23 and Figure 10.24 compare the results of access time for both queries (Q3 and Q4) in eleven different sizes of XML databases with and without Trust Based Access Control. The time consumed for Trust Based Access Control is longer than the time consumed without it for both queries. In the simple query, the time for Trust Based Access Control increases markedly when the size of the files is increased. The normal time to access the 0.027MB file is 31 milliseconds while the time consumed for Trust Based Access Control for the same file is 49 milliseconds. Then the time for Trust Based Access Control increases sharply to reach 3,863 milliseconds when the file size is 30.2MB. This time is almost double the normal time to access the 30.2MB file.

Using the complex query, the time starts at 50 milliseconds when the file size is 0.027MB and then reaches 4,076 milliseconds when the file size is 30.2MB. As expected, the access time without Trust Based Access Control is shorter. When the file size is 0.027MB, it needs around 32 milliseconds. Then the time increases gradually to reach 1,971 milliseconds when the file size is 30.2MB.

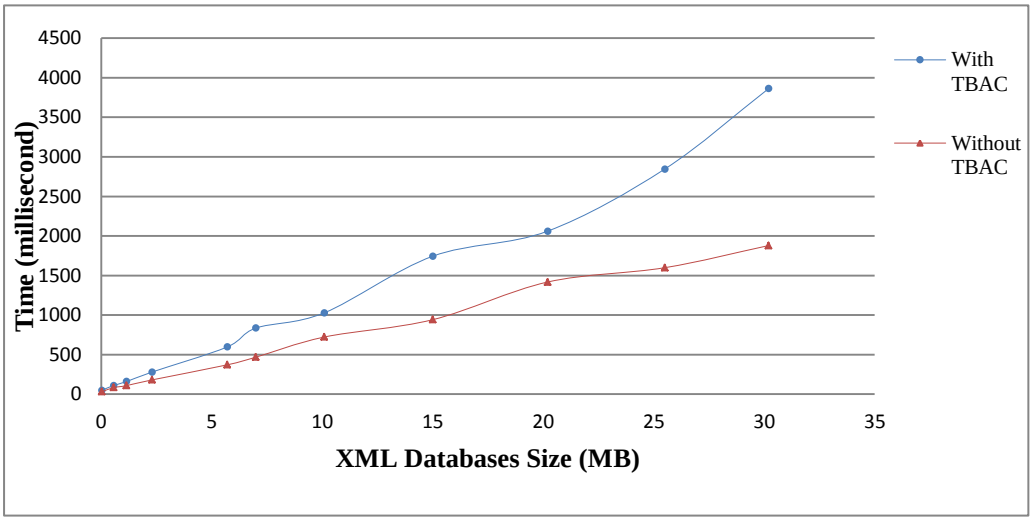


Figure 10.23 The comparative results for with and without TBAC in the simple query (Q3)

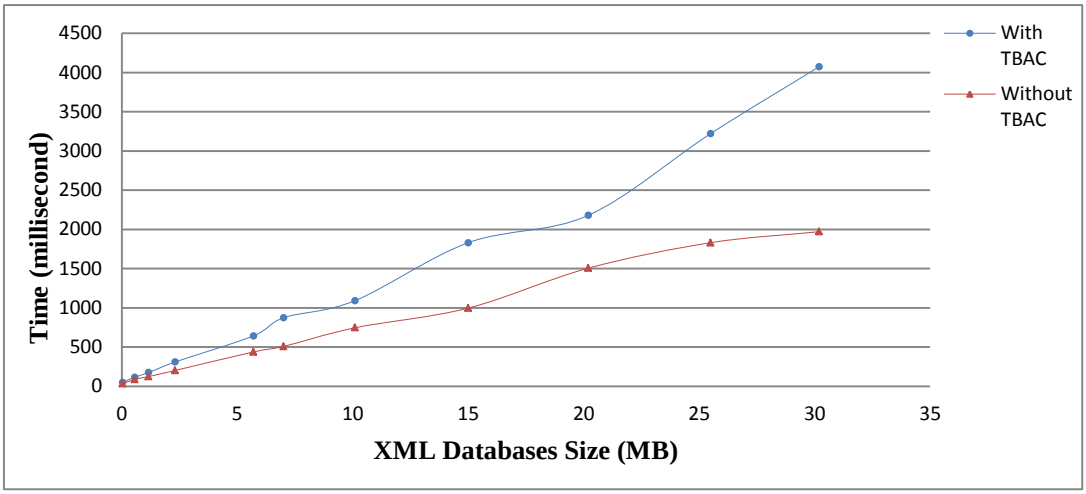


Figure 10.24 The comparative results for with and without TBAC in the complex query Q4

This experiment concludes that the time consumed for Trust Based Access Control is around 1.5 to 2 times the normal access time: Time for TBAC/time without TBAC = 1.5 ~ 2. The experimental work suggests that the Trust Based Access Control approach, though costly, is not prohibitively time consuming.

10.7.4 Conclusion

This experiment measured the real time required to use Trust Based Access Control and compared this with the normal access time. The experimental results showed that the time consumed for Trust Based Access Control increased markedly when the size of XML databases was increased. The results for both the simple and complex queries were similar. The comparison results of testing the access time with and without Trust Based Access Control system suggest it is worth persevering with this work. The performance time of TBAC is double the normal time, which is considered better than expected. This experiment showed that using TBAC consumed reasonable time while providing a secure environment through capturing misuse and updating privileges dynamically.

10.8 The Comparison with Mandatory Access Control (MAC) Experiment

This Section compares the performance speed of the proposed Trust Based Access Control system for XML databases with the traditional Mandatory Access Control for XML databases.

10.8.1 The Experimental Design Summary

The experiment aims to compare Trust Based Access Control and the Mandatory Access Control. As mentioned in Chapter 9, the Mandatory Access Control was selected for two main reasons. It is one of three main traditional access types. Using a well-known and well-established approach in the comparison makes the results fairer and more reasonable. The Mandatory Access Control approach is practically applied to XML databases and the experimental results were published by (Zhu et al., 2007; Zhu et al., 2009).

10.8.2 Analytic Procedures

Some results of the Trust Based Access Control with 1,000 users that were described in the previous experiment (Section 10.7) are also used here. For comparison, the published results of Mandatory Access Control were provided by Zhu et al. (Zhu et al., 2007; Zhu et al., 2009). The comparison between two systems was run with six different databases' sizes that were generated by XMark benchmark. These sizes were selected for ease of comparison with the published experimental work in the Mandatory Access Control. Both systems were tested using the simple and complex queries. The simple query was [Q3: //site/open_auctions/open_auction/initial] and the complex query was [Q4: //listitem//keyword]. The comparative results of the performance speed are presented graphically in the next Section.

10.8.3 Results Analysis

In terms of security, Mandatory Access Control (MAC) is one of the main traditional types of access control that are well-established, as mentioned in Chapter 3. It manages the access process based on classifying both subjects and objects into security levels. This approach has two main principles: read-down and write-up. Although it provides a high level of security, it is not widely used due to difficulties in classifying security levels in organisations. It also suffers from limitations in data integrity concepts due to using the write-up principle.

On the other hand, Trust Based Access Control (TBAC) is considered to be one of the new types of access control (see Chapter 4). The access process depends on the trust management system that evaluates users' operations over time. It is unlike MAC, which is static; it provides dynamic access control that updates privileges based on Trust Value (TV). Trust Based Access Control (TBAC) protects data from misuse by both outsiders and insiders by capturing bad transactions and errors. In contrast, MAC, like other traditional types, can prevent misuse by outsiders but can only handle limited forms of insider threats. Trust Based Access Control provides automatic calculations for TV that make the classification process for users' privileges easier than MAC, where classification can not be automatic. Although TBAC provides many new security features, it needs further developments to improve its functionalities.

This experiment compared the performance between TBAC and MAC for XML databases in practice. To obtain accurate results, the same data sets and a similar environment that has most of the platform features were used. The platform used by TBAC was slightly different in terms of operating system. MAC was run in Windows XP but TBAC was run in Windows 7. This point may affect the reliability of this experiment to some extent.

The comparative results between the Trust Based Access Control and the Mandatory Access Control are displayed in Figure 10.25 and Figure 10.26. Figure 10.25 shows the results for the simple query (Q3) in six different XML databases. In general, the performance of Trust Based Access Control is markedly faster than the Mandatory Access Control. The time was 50 milliseconds for TBAC and 220 milliseconds for MAC when the size of the database is 0.027MB. The time increased significantly in Mandatory Access Control to reach 3,900 milliseconds when the XML database size was 7MB. On the other hand, the time consumed for Trust Based Access Control grows slowly while the size of the database increases until it reaches 844 milliseconds for 7MB.

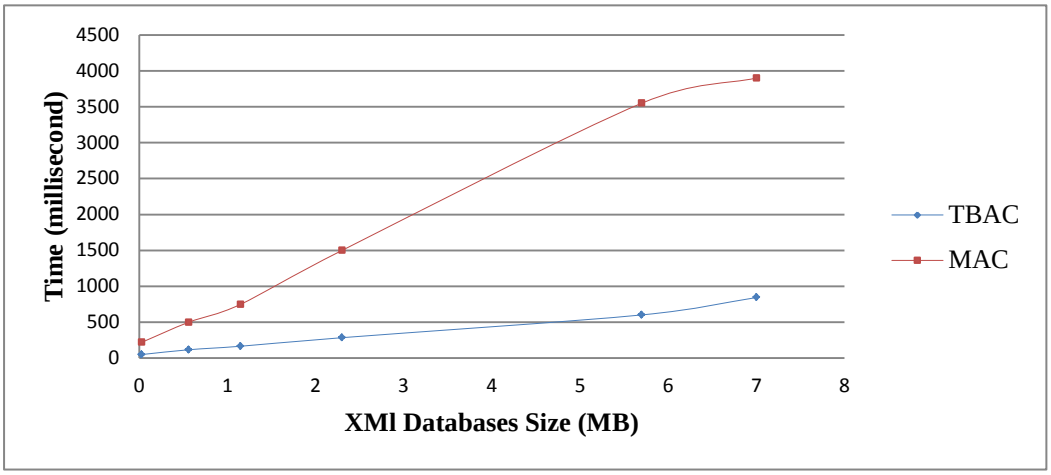


Figure 10.25 The comparative results between TBAC and MAC in the simple query (Q3)

Overall, the Trust Based Access Control requires less time than the Mandatory Access Control to run the complex query Q4. When the size of the XML database size was 0.027, the performance time in TBAC was 54 milliseconds and in MAC was 250 milliseconds. The time increases gradually in TBAC and significantly in MAC. The Trust Based Access Control requires approximately 887 milliseconds to access Q2 in the XML database with 7MB. In contrast, the Mandatory Access Control consumes more time, almost 4,300 milliseconds.

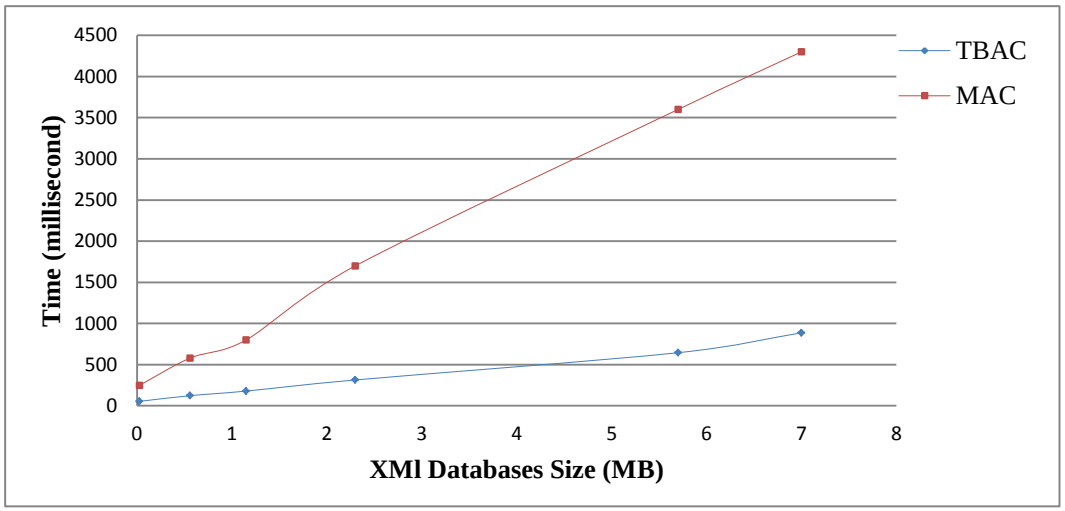


Figure 10.26 The comparative results between TBAC and MAC in the complex query Q4

The time consumed to perform a simple query was slightly faster than the complex query in both systems. The Trust Based Access Control performance speed is better than the traditional Mandatory Access Control because of the simple design structure of the Trust Based Access Control. As mentioned in Chapter 6, Trust Based Access Control divides the policy into many files and this improves the access time. The Trust Value (TV) for XML database is defined in a separate file called XML databases' access permission policy file.

In the Mandatory Access Control system developed by Zhu et al. (2007, 2009), the labels are added to the data in the original XML file that markedly increases the file size and makes the access process slow. More details about the storage consumption for the system is summarised in the Evaluation Chapter (Chapter 11).

10.8.4 Conclusion

This experiment compared the Trust Based Access Control for XML databases and the traditional Mandatory Access Control for XML databases. The results showed that the time consumed for Trust Based Access Control was

shorter than the Mandatory Access Control with all XML databases and both queries**.** The interesting comparative results suggest that the Trust Based Access Control added more security features with little time and storage cost; this was unexpected. The storage cost will be described in the next Chapter.

10.9 Conclusion

The experimental results were presented in this Chapter. Each experiment was discussed in detail and the main outcomes identified in its specific Section. The results of the first experiment displayed the flexibility in the calculation process of Trust Value. The second experiment showed that the writing and reading in the XLog file were affected by the number of recorded processes regardless of the type of process. The third experiment measured the access time in the access control module and identified that the total time depended on the retrieval time. The results of the trust maintenance experiment illustrated that this process needed little time due to the dynamic design of the XLog file. The fifth experiment explained the time consumed to run the access supervision process when the access is permitted and denied. The results showed that the process is faster when the access is denied because it did not include the retrieval time. The process time increased when the access was permitted and varied depending on the size of the XML databases. The comparative results in the sixth and seventh experiments showed that the Trust Based Access Control consumed a reasonable time while adding more security features compared to existing techniques.

In general, the results were acceptable. The evaluations for these results and the limitations of the experimental design are described in the next Chapter. Chapter 11 revisits the research hypotheses and summarises the main findings of applying the Trust Based Access Control for XML databases.

11 EVALUATION

11.1 Introduction

Chapter 10 provided and analysed the results of the experiment. In this Chapter, the system in general is evaluated (Section 11.2) and the experimental design and the results for specific experiments are evaluated individually (Section 11.3). The storage consumed by applying Trust Based Access Control for XML databases is also evaluated in Section 11.4. After the evaluation, the features and the limitations of experiments are summarised in Section 11.5. Section 11.6 contains the conclusions.

11.2 The System's Overall Evaluation

This Section describes the evaluation of the Trust Based Access Control for XML databases. The system was proposed based on the research hypothesis stated in Chapter 5 (Section 5.3):

"Since the Trust Based Access Control (TBAC) approach has been applied successfully in many areas, such as networks and virtual organisations, it may also improve security in the XML database research field by providing security to protect sensitive and confidential data from misuse by both outsiders and insiders while not restricting appropriate access."

This approach aimed to improve the security of XML databases as mentioned in the Section on the research objectives and contribution (Section 5.4). It intended to improve the security level in XML databases by providing a new dynamic access control for the XML database environment that depends on trust. The trust access control aimed to evaluate user's behaviour over time and

update privileges automatically. The improvement in the user performance could be considered as a side effect of using this technique but it has not been tested in this thesis.

To test the research hypothesis, the Trust Based Access Control for XML databases was implemented successfully. The design of the system was described in Chapters 6, 7, and 8. The system performance was evaluated by developing seven different experiments. These experiments were designed to ensure that the Trust Based Access Control could be applied to XML databases. The experimental design was discussed in Chapter 9 and the empirical results were analysed in Chapter 10.

It is clear that the research hypothesis was supported by reasonable and expected results based on the proposal (Chapter 5), the system design (Chapters 6 to 8), the experimental design (Chapter 9), and the results (Chapter 10). The trust access control was applied and extensively tested on XML databases. The results met the objectives and expectations. In general, the Trust Based Access Control for XML databases worked properly, protected sensitive data, evaluated users' behaviours, and updated privileges automatically with reasonable performance time.

This Section aims to evaluate the overall system to identify further improvements. Although the system was implemented properly and the results were obtained as hoped, some further improvements can be made to develop the system's efficiency. From the system design perspective, the Trust Value (TV) depends only on the direct trust and ignores the indirect trust. This concept worked properly and the TV was calculated correctly. The TV could be extended to depend on more factors such as the time and place. As mentioned in Chapter 6, the indirect trust is not relevant in the databases area. However, indirect factors rules could be included the calculations as additional features. For example, the recommendations factor can be used by the administrator of the system. The calculation technique for the TV was simple, which made the

system implementation phase easier and the performance faster. This technique could be improved considering machine learning and probability. This point will be discussed further under the heading of future works (Chapter 12).

The assignment of TV to nodes in XML databases could be developed by identifying group nodes that have the same TV which reduce the time and storage consumed. This will also be discussed as future work.

The system was designed to evaluate the user behaviour by capturing errors and bad transactions. Detecting errors and bad transactions depended on policy rules. In this system there were five rules for capturing bad transactions and three for capturing the errors. They were designed for the initial stage to ensure the system's functionality and were simple and limited to basic transactions in XML databases. They could easily be extended to cover more complex rules. Many rules for errors or bad transactions can be defined depending on a schema. Moreover, the method for detecting errors and bad transactions can be extended with other techniques used to capture insider threats (Yi and Panda, 2003; Chinchani et al., 2005; Chagarlamudi et al., 2009). These viewpoints are considered as suggestions in the future work, which is described in Chapter 12.

From an implementation perspective, applying the Trust Based Access Control for XML databases worked perfectly. The system was tested with different real data sets (Treebank, NASA, SIGMOD Record) and synthetic benchmarks (XMark). The larger data sets such as DBLP were not used due to the limitations in the resources. The scalability test can be developed by providing better facilities. The user sets covered small (50), medium (100), and large (1,000) organisations. The results were enough to gather and analyse data. More user sets may be used to provide further analysis. The practical evaluations for experimental results are discussed in detail in the next Section.

11.3 The Experiments Evaluation

This Section evaluates the experimental design (see Chapter 9) and results (see Chapter 10). In general, all results were reasonable and acceptable, as expected, but further improvements can be made.

11.3.1 The Trust Module Experiment

This Section discusses the evaluation of the design and the results of the first experiment, which focused on calculating Trust Value (TV). The design of this experiment (see Chapter 9) was simple and met its objectives. The experimental results were appropriate and accurate. The Trust Value increased when there were no errors or bad transaction, otherwise it declined. The results show flexibility in calculating TV depends on six factors. The weights can be changed depending on organisation policy to provide more freedom. The calculations were repeated with different weights (the maximum, the minimum, the medium range) then the recommended weights were selected. The recommended weights were selected to keep the Trust Value (TV) dynamic with respect to other access rules.

This experiment can be easily extended to cover other factors in the calculations. Although, the experimental results were enough to make the recommendations, the experiment can be repeated with other various readings to find more results for scalability evaluation.

11.3.2 The Logging Experiment

This Section evaluates the design of the XLog file and the results for reading and writing processes. The dynamic design of the XLog file makes the performance of reading, writing, and other related processes fast and easy. The results, as expected, showed that reading consumed less time than writing. Both processes' performances were affected markedly by the number of recorded processes regardless of the type.

From all results, it appears that using logging to both detect insider threats and update privileges in the Trust Based Access Control worked well and was therefore worthwhile. Logging is an important topic in security but it is rarely discussed in XML databases' literature. This approach of using logging in Trust Based Access Control and its results adds useful information to the literature on XML databases security. This leads to the question whether to apply traditional logging to XML databases. This viewpoint will be discussed later in the future work (Chapter 12).

11.3.3 The Access Control Module Experiment

The evaluation for the design and results of the access control module are discussed here. The simple design technique (see Chapter 9) used to find and compare the Trust Value (TV) for both the user and the node worked accurately. The results in Chapter 10 showed that the access time was affected mainly by the retrieval time, which depended on the size of XML databases. Due to the resources and time limitation, the scalability test for the access control module was run with only three data sets and three user sets. This test could be repeated with various data sets and user sets to find more results.

11.3.4 The Trust Maintenance Experiment

This experiment was designed to test the performance speed for the trust maintenance. The frequency of the trust maintenance process is defined by the organisation policy. The results showed that the time consumed was very short with different user sets. It is recommended to perform this process very frequently, such as hourly. This makes the system dynamic and more responsive to user behaviour. The short time required is related to the short and dynamic structure of the XLog file.

11.3.5 The Access Supervision Experiment

This Section evaluates the design and results of the access supervision process. The experiment was performed in two cases where access was permitted or denied with two simple XPath queries. The first was considered a bad transaction and the second a normal transaction. The results demonstrated that the time consumed when access is denied is shorter than when the access is permitted. The access time when access was allowed is mainly affected by the size of XML databases.

The design worked well and the results were reasonable; however, further improvements can be made to obtain more accurate results. The XML query syntax could be executed as an XQuery rather than XPath. Although, XQuery is more complicated than XPath, it is widely used in XML applications. The scalability test could be extended to cover XML benchmarks. The experiment could be repeated with several sizes of benchmarks' data sets. Moreover, the access supervision could be tested using the query sets designed by benchmarks to obtain comparative results.

11.3.6 The Experiment to Determine the Cost of Trust Based Access Control (TBAC)

This experiment was designed to test the whole system with eleven data sets generated by XMark using two XPath queries. The first query is simple and the second is complex. The results showed that Trust Based Access Control ran in reasonable time. The time is affected by the size of the data sets. These results were compared with the normal access result without TBAC. The comparative results showed that the access time in Trust Based Access Control is around double the normal access time. Even though Trust Based Access Control required more time, it provided more security features in an acceptable time range.

The experimental design and results were respectable but they could include more developments. The scalability test was performed with eleven data sets starting from 0.027MB to reach 30.2MB. The size ranges were selected with respect to the resources limitations. The system performance could be tested with larger databases on faster machines. With regard to the data sets generated by XMark, the query sets could also be used to check the performance and obtain more comparative results.

11.3.7 The Comparison with MAC Experiment

This Section evaluates the comparison between the Trust Based Access Control and the Mandatory Access Control for XML databases. The results showed that the TBAC performed faster than traditional Mandatory Access Control. This experiment presented good comparative results. It seems that the Trust Based Access Control improved security in XML databases within a reasonable time range.

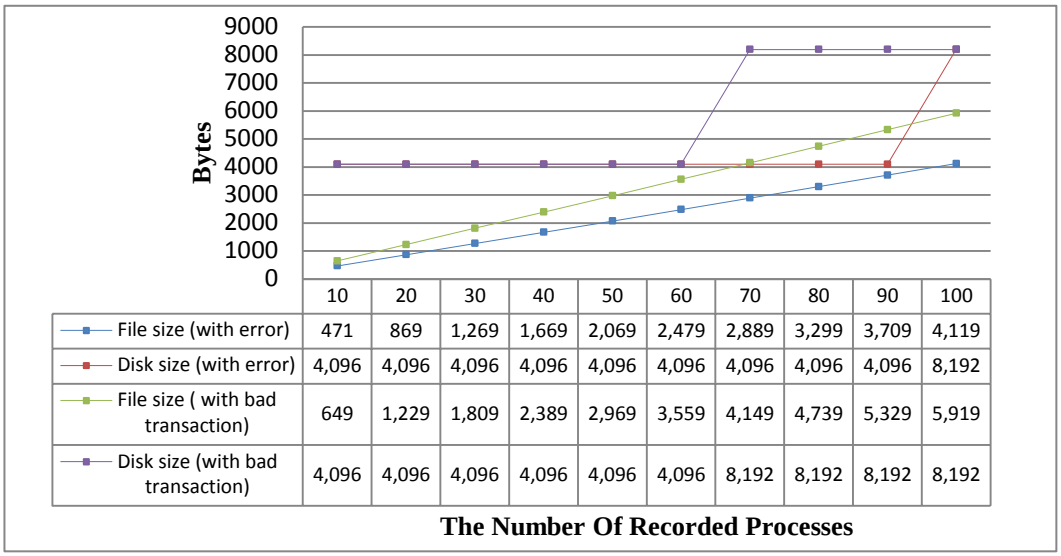
The comparison was limited to the Mandatory Access Control because it is implemented practically and has published results. Other traditional and new access approaches have been proposed but only theoretically without implementations. Implementing these approaches to obtain the results is beyond the scope of this work. It is possible that authors of other approaches may publish results in the future. It would be an interesting comparison.

11.4 The Storage Evaluation

Although this research concerns the security aspect in XML databases, this Section discusses storage in the Trust Based Access Control in general. The storage evaluation focuses on measuring space consumed to store the XML files. The Trust Based Access Control uses seven types of XML files, which include the XLog file, the error policy file, the operation policy file, the trust policy file, the XML database's permission policy file, the user's permission policy file, and

the original XML database. The structure, the syntax, and the contents of these files were described earlier in Chapters 7 and 8. This Section evaluates the file size, the disk size, and finds the total storage space required.

The XLog file size changes depending on the type and number of recorded transactions. Three types of the XLog files were evaluated: with errors only, bad transactions only, and a mixture of both. When the file contains errors only, the size increases gradually with the number of recorded errors. The file size starts at 471 Bytes with ten errors and grows to reach 4,119 Bytes with 100 errors. The disk size remains stable at 4,096 Bytes for all versions of the XLog file but the disk size doubles when the number of errors is 100. The file size consumes more storage space when the XLog file contains bad transactions only. The file size is 649 Bytes when the number of bad transaction is 10 and 5,919 Bytes when the number of bad transaction is 100. The disk size is 4,096 Bytes when the XLog file contains 10 to 60 bad transactions and 8,192 Bytes when the XLog file has 70, 80, 90, and 100 bad transactions. Figure 11.1 shows both the file size and the disk size for the XLog file when it has errors only or bad transactions only. The size of the XLog file that has both errors and bad transaction increases markedly when the number of process is increased. The file size is 1,094 Bytes when it has 10 errors and 10 bad transactions. It reaches 10,019 Bytes when the XLog file includes 100 errors and 100 bad transactions. The disk size keeps stable for three or four versions and then increases by 4,096 Bytes each time. Figure 11.2 illustrates the storage consumed for the XLog file including both errors and bad transactions.



	10	20	30	40	50	60	70	80	90	100
File size (with error)	471	869	1,269	1,669	2,069	2,479	2,889	3,299	3,709	4,119
Disk size (with error)	4,096	4,096	4,096	4,096	4,096	4,096	4,096	4,096	4,096	8,192
File size (with bad transaction)	649	1,229	1,809	2,389	2,969	3,559	4,149	4,739	5,329	5,919
Disk size (with bad transaction)	4,096	4,096	4,096	4,096	4,096	4,096	8,192	8,192	8,192	8,192

.

Figure 11.1 The file size and the disk size for the XLog file containing errors only or bad transactions only

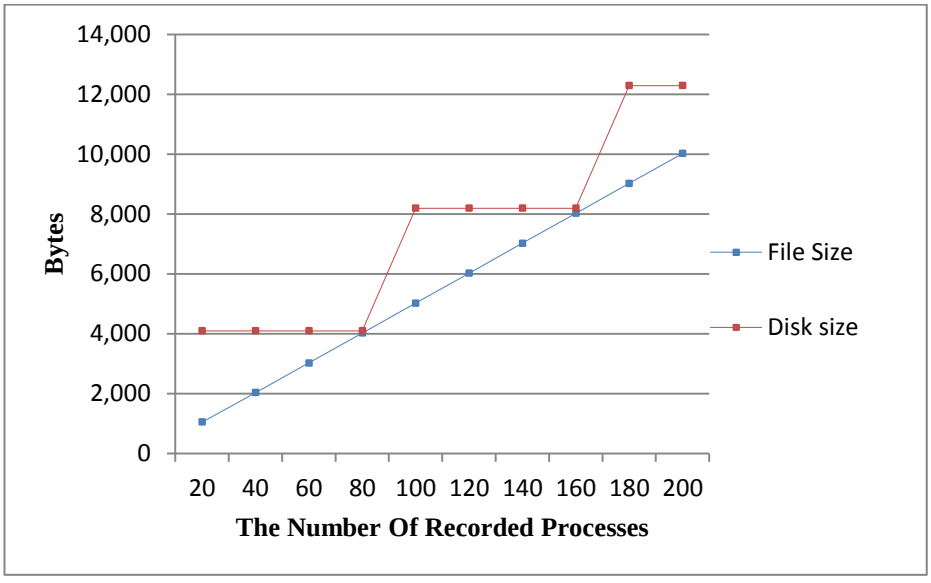


Figure 11.2 The file size and the disk size for the XLog file containing a mixture of errors or bad transactions

The three policy files are related to the trust module – the error policy file, the operation policy file and the trust policy file – were not tested as multiple versions unlike the XLog file, but they could easily be extended to cover additional policy rules. In general, these files consume little storage space. The file size for the error policy is 288 Bytes, for the operation policy is 486 Bytes, and for the trust policy is 955 Bytes. All three files require the same disk size 4,096 Bytes.

The users' access permission policy file was evaluated with three user sets: 50, 100, and 1,000. The storage required increased depending on the increase in the number of users. When the file includes 50 users, the file size is 5,368 Bytes and the disk size is 8,192 Bytes. This file size increases to 10,843 Bytes and the disk size to 12,288 Bytes with 100 users. When the number of users is 1,000, the file size is 107,146 Bytes and the disk size is 110,592 Bytes. The storage required for the users' permission policy file with three user sets is presented graphically in Figure 11.3.

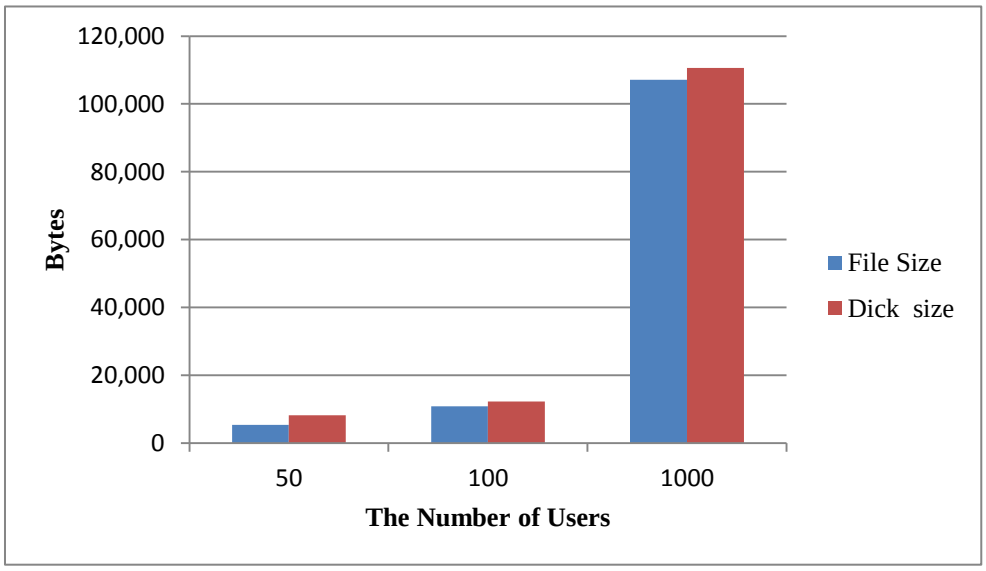


Figure 11.3 The file size and the disk size for the users' access permission policy file

The databases' permission policy file size is related to the structure of the XML database. The storage space consumed while applying the Trust Based Access Control with the three real-world data sets, were used in the experiments (see Chapter 9), is explained in Table 11.1. This table measures the file size in kilobytes (KB) for all XML files used in the system with 50 users and 20 recorded processes in the XLog file. Based on this table, Figure 11.4 compares the storage required for three data sets with their own and with TBAC to find the storage cost for using the Trust Based Access Control. This diagram shows that the storage space required to apply TBAC is small, around 9KB. The storage needed for the XML data sets with and without TBAC is almost the same.

Table 11.1 The file size storage consumed by applying TBAC for the selected real-world data sets

The name of file	SIGMOD Record	NASA	TreeBank
The original XML file	466	24,464	84,065
The XML database's permission policy file	0.571	1.32	0.815
The user's access permission policy file (50)	5.242	5.242	5.242
The error policy file	0.281	0.281	0.281
The operation policy file	0.474	0.474	0.474
The trust policy file	0.932	0.932	0.932
The XLog file (20)	1.024	1.024	1.024
Total size	474.524 (KB)	24,473.273 (KB)	84,073.768 (KB)

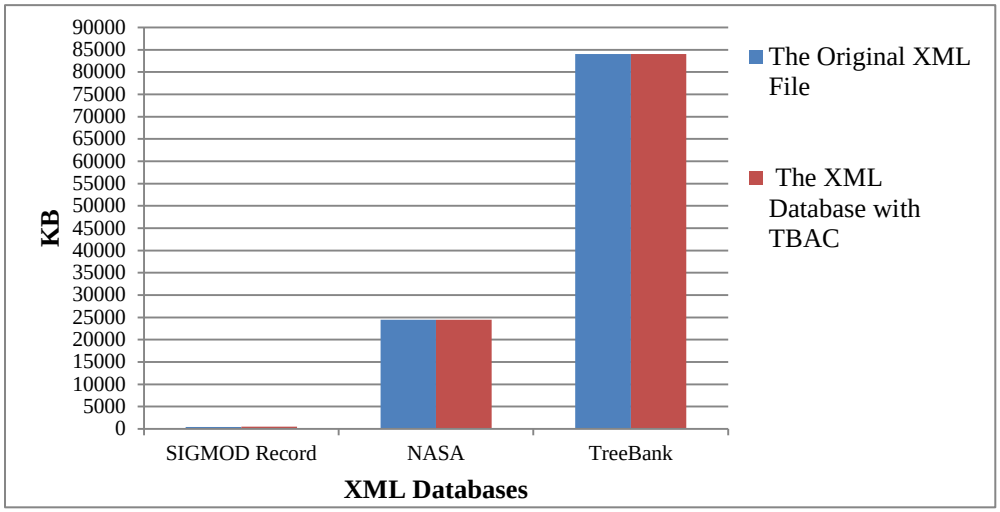
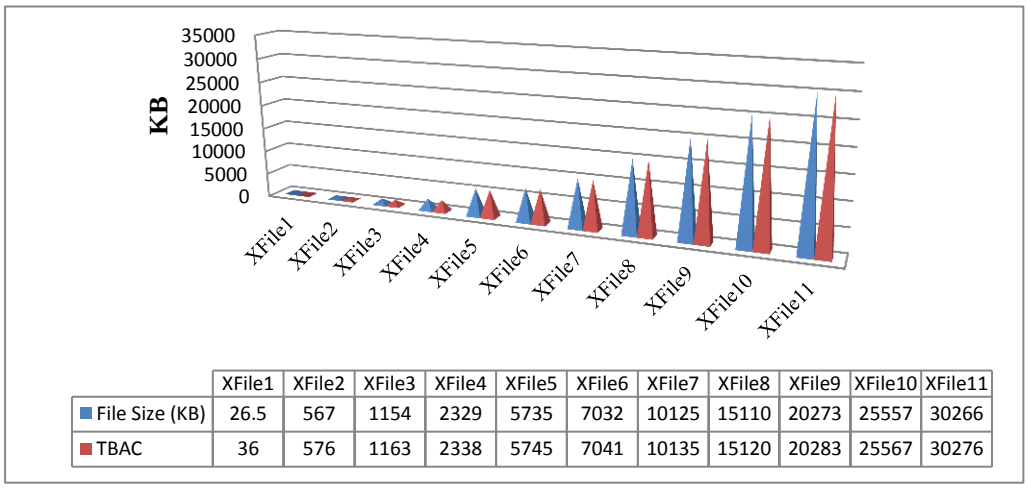


Figure 11.4 The comparison between the storage consumed with and without TBAC for three selected real data sets

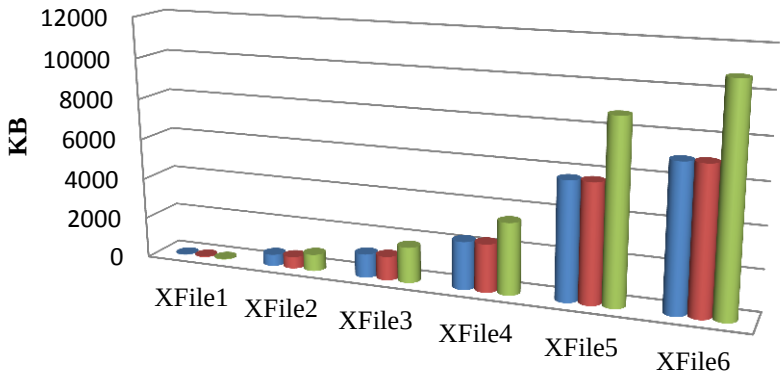
The storage consumed by Trust Based Access Control with the eleven data sets generated by the XMark (see Chapter 9), are illustrated in Figure 11.5. The total file sizes for each data set with TBAC are larger than the original file size by around 10 KB. The total file sizes are the summation of the XML files (seven types) used in TBAC.



	XFile1	XFile2	XFile3	XFile4	XFile5	XFile6	XFile7	XFile8	XFile9	XFile10	XFile11
File Size (KB)	26.5	567	1154	2329	5735	7032	10125	15110	20273	25557	30266
TBAC	36	576	1163	2338	5745	7041	10135	15120	20283	25567	30276

Figure 11.5 The comparison between the storage consumed with and without TBAC for XMark data sets

To compare the storage required for Trust Based Access Control (TBAC) and Mandatory Access Control (MAC), only six data sets out of elven were used to match the published material. As mentioned earlier, the storage required for the Trust Based Access Control includes the file size of the XLog file, the error policy file, the operation policy file, the trust policy file, the users' access permission policy file, the database's access permission policy file, and the XML data set file. The Trust Based Access Control approach breaks down policies into many files to avoid repetition in assigning TV for data. This technique saves much storage space. In contrast, Mandatory Access Control (MAC) includes the access policy in the XML data set file. This increases the storage space required for MAC as assigning labels to data is repeated. In general, the storage space required to handle the data set in Mandatory Access Control is larger than in Trust Based Access Control. Figure 11.6 shows the storage space for the normal XML data sets, with both TBAC and with MAC.



	XFile1	XFile2	XFile3	XFile4	XFile5	XFile6
File Size (KB)	26.5	567	1154	2329	5735	7032
TBAC	36	576	1163	2338	5745	7041
MAC	41.984	819.2	1740.8	3481.6	8806	10752

Figure 11.6 The comparison between the storage consumed with TBAC and MAC for XMark data sets

11.5 The Features, Limitations, and The Main Findings of The Experiments

Although the seven experiments were intended to be simple, accurate, and effective, they had limitations. The features of these experiments are described in Section 11.5.1; the limitations are highlighted in Section 11.5.2.

11.5.1 Features of the Experiments

In order to meet the objectives below, seven experiments were designed to cover a number of aspects. The objectives were:

- to implement the Trust Based Access Control for XML databases in real environments and find the system's actual performance.
- to test the system over several XML databases. The data sets were selected carefully to include different structures such as depth. Both natural and synthetic data sets were used.
- to include comprehensive view points to test XML databases such as functionality, performance, and scalability.
- to provide the experimental design and results to be published and available for further and related research.

11.5.2 Limitations of The Experiments

Although the experiments worked accurately, they suffer from some limitations. The experimental design was simple and covered only basic aspects. The theory behind this was to ensure that the system works properly first before extending it to cover more complex features. They could be extended to obtain more elaborate results by using more data sets, user sets, and query sets generated by benchmarks. The limitation on the XML database size due to the need to keep it in memory could be improved through using a larger computer but it will always be a restriction. Many technical points, which were described in detail in the previous Sections, could be used to improve and extend the

system as future work (Chapter 12). These cover calculation techniques, the rules for detecting errors and bad transaction, the method for capturing insider threats and the access queries language.

11.5.3 The Main Findings of the Experiments

The important finding is that the experiment results confirmed the main idea in the research hypothesis. In general, all the experiments showed that the Trust Based Access Control can be applied practically to XML databases. They demonstrated the flexibility in calculating Trust Value, simplicity in capturing errors and bad transaction, regularity in the access process, and scalability with different data sets and user sets. Each experiment met its objectives. The results of the experiments can be summarised by saying that the Trust Based Access Control provides more security features with reasonable time and storage. The main findings of the whole research are described in the next Chapter.

11.6 Conclusion

Initially, in order to develop the system, some simple rules were applied to ensure that the system worked perfectly, before any further developments were made. The Trust Based Access Control has been applied to XML databases but it can be extended to cover further developments. This Chapter evaluated the experiments and their results. Limitations in the system and experiments were identified. Some suggestions and improvements were mentioned briefly. They are explained in more detail in the future work Section in the next Chapter.

12 CONCLUSION AND FUTURE WORK

12.1 Introduction

This work attempted to improve security in XML databases using dynamic trust access control. The objectives, design, experiments, results, and evaluation were described in the previous Chapters. This is the final Chapter of the thesis. Section 12.2 summarises the work completed in relation to this research. The main contributions of the research are highlighted in Section 12.3; it relates the outcomes to the hypothesis in 12.4. Further development and future work are discussed in Section 12.5.

12.2 Thesis Summary

Based on the system development life cycle (SDLC) (Avison and Fitzgerald, 2006) that is widely used in system engineering and information systems, this research was designed to provide more security features for XML databases. The objective was to improve security in XML databases using dynamic access control depending on trust. These objectives were defined in Chapter 5. XML databases are an active research topic due to a recent increase in their use (Abiteboul et al., 2000; Champion, 2001; Vakali et al., 2005; Sun and Wang, 2011; Noaman and Almansour, 2012; Verma et al., 2012; Desai, 2013; Vela et al., 2013). As with any database, they can contain sensitive and important data; therefore, it is imperative to be able to provide a secure environment to deal with such data. Secure systems need access control to manage access to the data and prevent unauthorised and malicious processes.

Traditional access models are limited in that they are static and focused mostly on protection from outsiders (Xing et al., 2010; Sun and Wang, 2011). Access control is one of the main issues in XML databases that need further

investigation (Sun and Wang, 2011; Verma et al., 2012; Desai, 2013). There has been extensive research in this area, discussed in Chapter 3, but many points still need to be developed. At present there is no golden standard for access control in XML databases (Lee and Yu, 2008; Gonzalez et al., 2009).

Trust Based Access Control is an established technology and is used in many areas, such as networks and virtual organisations (Almenarez et al., 2006; Lin et al., 2006; Ma et al., 2008; Xing et al., 2010; Singh, 2011). It depends on a trust management system, which automatically calculates and updates the Trust Values of users. Trust Values rely on users' behaviour, history, credit, and operations. Users can access resources for which their Trust Value is appropriate.

The Trust Based Access Control tracks users' errors and bad transactions over time and updates their privileges dynamically. It prevents outsiders' attacks as well as insiders' malicious processes, effectively preventing users from taking advantage of their role within certain limits.

Chapter 1 outlined the thesis structure and listed the published work. Chapter 2 described the main topics in XML databases such as components, syntax, and parsing. Chapter 3 explained security in XML databases and discussed the access control approaches and methods used in current research. Access control models for XML databases can be categorised into the Discretionary Access Control model (DAC), the Mandatory Access Control model (MAC), and Role Based Access Control (RBAC) (Hitchens and Varadharajan, 2001; Wang and Osborn, 2004; Zhu et al., 2007; Zhu et al., 2009; Rashid et al., 2010; Sun and Wang, 2011). The related work on Trust Based Access Control was then discussed in Chapter 4. Chapter 5 described the research motivations and objectives and highlighted the hypothesis.

The design stage includes Chapters 6, 7, and 8. The overall structure of Trust Based Access Control was described in Chapter 6. The system consists of two modules: the trust module and the access control module. It focuses on

observing users' behaviour by recording and evaluating bad transactions and errors. The access control module aims to make access decision depending on the access policies. Each of these two modules contains many components.

The trust module includes an operation recorder, an error detector, an operation evaluator, and a trust calculator. The operation recorder registers both errors and bad transactions in an XLog file. The error detector and the operation evaluator depend on policy files that define what an error or a bad transaction is. The trust calculator calculates the new Trust Value depending on three factors: existing Trust Value, Error Factor, and Bad Transaction Factor. The trust module was explained in detail in Chapter 7.

The access module consists of the access manager, which works in the light of access policies files and the access decision maker, and is responsible for handling queries. It either permits or denies the request. The access control module was explained in Chapter 8.

After designing the system, it was implemented. Implementation was tested using seven experiments that included several tools, data sets, and user sets. Chapter 9 showed the experimental design and Chapter 10 illustrated the practical results of applying Trust Based Access Control for XML databases.

Then the evaluation of the implementation included the experimental design and results. It can be summarised by saying that applying Trust Based Access Control for XML databases consumed reasonable time and storage. The system is important in that it tackles security problems in XML databases by reducing misuse and the cost is not excessive. The following Section discusses the research contributions.

12.3 The Main Contributions of the Research(Findings)

This research addressed the security problems in XML databases and offered a solution. It aimed to take advantage of trust approach, which is used in other areas, and applied it to XML databases. This thesis and published work contributed to the literature on the following main points:

- A new dynamic access control type for XML databases depending on a trust factor that evaluates user behaviours over time and updates the privileges.
- The research provided more security for XML databases.
- The research can capture misuse from both insiders and outsiders and so improved the security level for handling XML databases.
- The research offered empirical proof of Trust Based Access Control for XML databases.

12.4 Relating Research Outcomes to Hypothesis

The research hypothesis in Chapter 5 proposed that the Trust Based Access Control can be applied to improve the security of XML databases. As can be seen from experimental results and evaluation, the trust access control approach was successfully applied to XML databases with reasonable cost in terms of time and storage.

The hypothesis was tested with seven experiments using several data sets and user sets. Interesting results were found that showed the flexibility, simplicity, and efficiency of using Trust Based Access Control for XML databases. The first experiment showed the flexibility in calculating Trust Value based on several factors. The second experiment developed the XLog file that was used to record bad transactions and errors. The third experiment explained and measured the access process. The fourth and fifth experiments tested the whole system and measured the time consumed to perform the trust maintenance and the access supervision. The sixth and seventh experiments found the real

cost of using trust access control in XML databases and compared it with normal access and the Mandatory Access Control. All results supported the hypothesis and provided worthwhile information regarding security in XML databases.

12.5 Future Work

Trust Based Access Control is a relatively new approach in security. Applying this type of access control to XML databases is a novel idea proposed in this thesis. The research work in the thesis covered many points but still there is plenty of scope for further investigation.

More complex calculation techniques could be used to calculate the Trust Value. The trust calculator could depend on numerous approaches in machine learning and probability such as Bayes' theorem (Almenarez et al., 2006). The way the Trust Value is assigned to the node could also be improved to reduce time consumed and storage space. The Trust Value could be assigned to a group of nodes rather than just one node using techniques that are used in compression XML databases (Müldner et al., 2009).

The rules for defining errors and bad transactions could be extended to capture other problems. One way to extend the rules is to use a schema to find changes in the XML databases' structure. If a transaction would cause damage to the schema structure, it could be classified as a bad transaction. If the transaction could affect the structure, it might be defined as an error.

Beside the rules, the method used to capture bad transactions and errors could also be improved based on the techniques used for insider threats. Insider threat is a huge topic in security and different approaches have been proposed to solve this problem. This topic included tackling the transactions sequence as a way of capturing misuse (Yi and Brajendra, 2003; Chagarlamudi et al., 2009) or using read set, pre-write set, and post-write set, and then checking these sets (Yi and Brajendra, 2003). Some approaches define a normal scenario to perform

each task, then record the user scenario and compare it to the normal one (Chinchani et al., 2005). The system in this thesis could possibly be extended with these techniques to detect misuse.

Logging is one of the important topics in security but it is rarely discussed in XML databases. In this research, it is used to support the access control and dynamic update for privileges. Implementing the XLog file showed interesting results and this leads to the possibility of also applying traditional types of log for XML databases.

The Trust Based Access Control for XML databases was implemented for the first time as part of this research. To use it for existing systems that already have their own access control system may face some obstacles. So, rather than replacing the existing approaches, it could be integrated with them to provide hybrid access control that includes dynamic update to privileges. It could also be applied to existing systems as a top security layer that offers more security features without much extra cost.

As explained in this thesis, the Trust Based Access Control was applied successfully for XML databases with reasonable time and storage range. The flexible and hierarchy structure of XML databases supports and fits smoothly with the Trust Based Access Control approach. It could possibly be applied to other types of databases as well. Applying Trust Based Access Control to relational database is one of the main areas for further investigation.

12.6 Finally

This research focused on security in XML databases. It developed a dynamic access control for XML databases based on trust. This new approach evaluated users' behaviour over time and updated their privileges. It improved security and detected misuse from insiders and outsiders. This Chapter summarised the work done and highlighted the research outcomes. Sometimes

answering a question will raise many other questions; as happened with this thesis. Due to time limitations, these questions and other developments for the Trust Based Access Control for XML databases will have to be further investigated in the future.

13 REFERENCES

Abd El-Aziz, A. & Kannan, A. 2012a. Access Control for Healthcare Data Using Extended Xacml-Srbac Model The International Conference on Computer Communication and Informatics (ICCCI) Coimbatore, INDIA: IEEE, 1-4.

Abd El-Aziz, A. & Kannan, A. 2012b. Storing Xml Document and Xml Policies in Relational Databases. The International Conference on Computer Communication and Informatics (ICCCI) Coimbatore, INDIA: IEEE, 1-7.

Abd El-Aziz, A. & Kannan, A. 2012c. Storing Xml Rules in Relational Storage of Xml Dtd. The Second International Conference on Computational Science, Engineering and Information Technology. , USA: ACM, 408-412.

Abdul-Rahman, A. & Hailes, S. 1997. A Distributed Trust Model. In Proceeding of the ACM Workshop on New Security Paradigms, United Kingdom, 46-60.

Abiteboul, S. 1999. On Views and Xml. *ACM SIGMOD Record*, 28, 30-38.

Abiteboul, S., Buneman, P. & Suciu, D. 2000. *Data on the Web: From Relations to Semistructured Data and Xml*. California, Morgan Kaufmann Publishers.

Al-Badawi, M. 2010. *A Performance Evaluation of a New Bitmap-Based Xml Processing Approach.* PhD Thesis, The University of Sheffield.

Al-Shaikh, R., Hashim, G., Binhuraib, A. & Mohammed, S. 2010. A Modulo-Based Labeling Scheme for Dynamically Ordered Xml Trees. Fifth International Conference on Digital Information Management (ICDIM), 213-221.

Almenarez, F., Marin, A., Campo, C. & Carlos, R. 2005. Trust Ac: Trust-Based Access Control for Pervasive Devices. *LNCS*, 3450, 225-238.

Almenarez, F., Marin, A., Campo, C. & Garcia, C. 2004. Ptm: A Pervasive Trust Management Model for Dynamic Open Environments. First Workshop on Pervasive Security,Privacy and Trust PSPT04 in conjunction with Mobiquitous 2004.

Almenarez, F., Marin, A., Diaz, D. & Sanchez, J. 2006. Developing a Model for Trust Management in Pervasive Devices. Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, 52-71.

Amagasa, T., Yoshikawa, M. & Uemura, S. 2003. Qrs: A Robust Numbering Scheme for Xml Documents. 19th International Conference on Data Engineering, 705-707.

An, D. & Park, S. 2007. New Access Control for Secure Query Processing over Xml Data Stream. International Conference on Convergence Information Technology, 1764-1763.

An, D. & Park, S. 2011. Efficient Access Control Labeling Scheme for Secure Xml Query Processing. Computer Standards and Interfaces, 439-447.

Anderson, T. 2008. *Use an Xml Database in Php and Java Applications* [Online]. IBM. Available: http://www.ibm.com/developerworks/xml/tutorials/x-xmldbjavaphp/index.html [Accessed 21-5-2013].

Ardagna, C., Damiani, E., Vimercati, S. D. C. D. & Samarati, P. 2007. Xml Security. *In Security,Privacy and Trust in Modern Data Management*, 6, 71-86.

Arion, A., Benzaken, R., Manolescu, I. & Papakonstantinou, Y. 2007. Structured Materialized Views for Xml Queries. Proceedings of the 33rd International Conference on Very Large Databases, Vienna, Austria, 1325865: VLDB Endowment, 87-98.

Arion, A., Bonifati, A., Costa, G., Daguanno, S., Manolescu, I. & Pugliese, A. 2004. Efficient Query Evaluation over Compressed Xml Data. *Lecture Notes in Computer Science (LNCS),* 2992/2004, 637-638.

Avison, D. & Fitzgerald, G. 2006. *Information Systems Development: Methodologies, Techniques and Tools.* UK, McGraw-Hill Education.

Azzedin, F. & Maheswaran, M. 2002. Evolving and Managing Trust in Grid Computing Systems Canadian Conference on Electrical and Computer Engineering (CCECE) Canada, 1424-1429.

Balmin, A., Özcan, F., Beyer, K. S., Cochrane, R. J. & Pirahesh, H. 2004. A Framework for Using Materialized Xpath Views in Xml Query Processing. The Thirtieth International Conference on Very Large Databases, Toronto, Canada, 60-71.

Berglund, A., Boag, S., Chamberlin, D., Fernández, M., Kay, M., Robie, J. & Siméon, J. 2010. *Xml Path Language (Xpath) 2.0 (Second Edition)* [Online]. Available: http://www.w3.org/TR/xpath20/ [Accessed 18-3-2013].

Bertino, E., Braun, M., Castano, S. & Ferrari, E. 2000. Authorx:A Java-Based System for Xml Data Protection. The 14th Annual IFIB WG 11.3 Working Conference on Database Security, Netherlands, 1-19.

Bertino, E. & Sandhu, R. 2005. Database Security - Concepts, Approaches, and Challenges. *IEEE Transactions on Dependable and Secure Computing,* 2, 2-19.

Bhatti, R., Bertino, E. & Ghafoor, A. 2004. A Trust-Based Context-Aware Access Control Model for Web-Services. IEEE International Conference on Web Services, 184-191.

Bin, M. & Shijin, Y. 2010. A Method for Evaluating Initial Trust Value of Direct Trust and Recommender Trust. 2010 International Conference on Computer Design and Applications (ICCDA), V2-185-V2-190.

Boag, S., Chamberlin, D., Fernandez, M. F., Florescu, D., Robie, J. & Simacon, J. 2011. *Xquery 1.0: An Xml Query Language (Second Edition)* [Online]. W3C. Available: http://www.w3.org/TR/xquery/ [Accessed 15-05-2013].

Bobba, R., Fatemieh, O., Khan, F., Khan, A., Gunter, C. A., Khurana, H. & Prabhakaran, M. 2010. Attribute-Based Messaging: Access Control and Confidentiality. *ACM Trans. Inf. Syst. Secur.*, 13, 1-35.

Böhme, T. & Rahm, E. 2000. *Xmach-1: A Benchmark for Xml Data Management* [Online]. Available: http://dbs.unileipzig.de/en/projekte/XML/paper/XMach-1.html [Accessed 18-03-2013].

Böhme, T. & Rahm, E. 2003. Multi-User Evalaution of Xml Data Management Systems with Xmach~1. VLDB 2002 Workshop EEXTT and CAiSE London, UK, 148-159.

Bourret, R. 2005. *Xml and Databases* [Online]. Available: http://www.rpbourret.com/xml/XMLAndDatabases.htm [Accessed 06-10-2011].

Brownell, D. 2002. *Sax2*. USA, O'Reilly and Associates Publishers.

Byun, C. & Park, S. 2010. A Schema Based Approach to Valid Xml Access Control. *Information Science And Engineering*, 26, 1719-1739.

Byun, J.-W., Bertino, E. & Li, N. 2005. Purpose Based Access Control of Complex Data for Privacy Protection. The Tenth ACM Symposium on Access Control Models and Technologies, Stockholm, Sweden, 1063998: ACM, 102-110.

Cahill, V., Gray, E., Seigneur, J. M., Jensen, C. D., Yong, C., Shand, B., Dimmock, N., Twigg, A., Bacon, J., English, C., Wagealla, W., Terzis, S., Nixon, P., Di Marzo Serugendo, G., Bryce, C., Carbone, M., Krukow, K. & Nielson, M. 2003. Using Trust for Secure Collaboration in Uncertain Environments. *Pervasive Computing, IEEE*, 2, 52-61.

Cameron, D. 2008. *How Xquery Extends Xpath* [Online]. IBM. Available: http://www.ibm.com/developerworks/xml/library/x-queryxpath/index.html [Accessed 15-05-2013].

Carey, M., Dewitt, D. & Naughton, J. 1993. The Oo7 Benchmark. *ACM/SIGMOD Record*, 22, 12-21.

Chagarlamudi, M., Panda, B. & Yi, H. 2009. Insider Threat in Database Systems: Preventing Malicious Users' Activities in Databases. Sixth International Conference on Information Technology: New Generations 1616-1620.

Champion, M. 2001. Storing Xml in Databases. *EAI Journal*, 10, 53-55.

Chan, S. S. M., Qing, L. & Pino, J. A. 2004. Access Control Mechanism for Collaborative Video Database Production Applications. IEEE Sixth International Symposium on Multimedia Software Engineering 396-402.

Chase, N. 2003a. *Understanding Relax Ng* [Online]. IBM. Available: http://www.ibm.com/developerworks/xml/tutorials/x-relaxng/index.html [Accessed 05-11-2012].

Chase, N. 2003b. *Validating Xml* [Online]. IBM. Available: http://www.ibm.com/developerworks/xml/tutorials/x-valid/section4.html [Accessed 05-11-2012].

Chen, S., Li, H., Tatemura, J., Hsiung, W., Agrawal, D. & Candan, K. S. 2006. Twig2stack: Bottom up Processing of Generalized Tree Pattern Queries over Xml Documents. VLDB Seoul, Korea., 283-294.

Chen, T., Lu, J. & Ling, T. W. 2005. On Boosting Holism in Xml Twig Pattern Matching Using Structural Indexing Techniques. SIGMOD Baltimore, Maryland, USA, 455--466.

Chen, Y. B., Ling, T. W. & Lee, M.-L. 2002. Designing Valid Xml Views. The 21st International Conference on Conceptual Modeling, London, UK, 463-478.

Chin, S. & Older, S. 2011. *Access Control, Security, and Trust a Logical Approach.* USA, Chapman & Hall / CRC.

Chinchani, R., Iyer, A., Ngo, Q. & Upadhyaya, S. 2005. Towards a Theory of Insider Threat Assessment. International Conference on Dependable Systems and Networks, USA: IEEE Xplore, 108-117

Cho, S., Yahia, S., Lakshmanan, L. V. S. & Srivastava, D. 2002. Optimizing the Secure Evaluation of Twig Queries. The 28th Very Large Databases Conference (VLDB), China, 490-501.

Cohen, E., Kaplan, H. & Milo, T. 2002. Labeling Dynamic Xml Trees. Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, Madison, Wisconsin, 543648: ACM, 271-281.

Damiani, E., De Capitani Di Vimercati, S., Stefano, P. & Samarati, P. 2000a. Design and Implementation of an Access Control Processor for Xml Documents. *Computer Networks,* 33, 59-75.

Damiani, E., Di Vimercati, S. D. C., Paraboschi, S. & Samarati, P. 2000b. Securing Xml Documents. The 2000 International Conference on Extending Database Technology (EDBT), New York: Springer.

Damiani, E., Di Vimercati, S. D. C. & Samarati, P. 2005. New Paradigms for Access Control in Open Environments. The Fifth IEEE International Symposium on Signal Processing and Information Technology, 540-545.

Damiani, E., Fansi, M., Gabillon, A. & Marrara, S. 2007. Securely Updating Xml. KES 2007, Italy.

Damiani, E., Fansi, M., Gaillon, A. & Marrara, S. 2008. A General Approach to Securely Querying Xml. *Computer Standards & Interfaces,* 30, 379-389.

Damiani, E., Samarati, P., De Capitani Di Vimercati, S. & Paraboschi, S. 2001. Controlling Access to Xml Documents. *Internet Computing, IEEE,* 5, 18-28.

Damiani, E., Vimercati, S. D. C. D., Paraboschi, S. & Samarati, P. 2002. A Fine-Grained Access Control System for Xml Documents. *ACM Transactions on Information and System Security (TISSEC),* 5, 169-202.

Darugar, P. 2000. *Dare to Script Tree-Based Xml with Perl* [Online]. IBM. Available: http://www.ibm.com/developerworks/xml/library/xml-perl2/index.html [Accessed 05-11-2012].

Davis, K., Zhan, Y. & Davis, R. 2003. An Xml/Xpath Query Language and Xmark Performance Study. The 2003 Symposium on Applications and the Internet, Orlando, FL, USA, 422-427.

Dblp. 2013. *The Dblp Computer Science Bibliography* [Online]. Available: http://dblp.uni-trier.de/db/ [Accessed 18-03-2013].

Desai, A. 2013. Xml Security Using DNA Technology. *International Journal of Engineering Research & Technology,* 2, 1-6.

Deschler, K. & Rundensteiner, E. 2003. Mass: A Multi-Axis Storage Structure for Large Xml Documents. The twelfth international Conference on Information and Knowledge Management USA: ACM, 520-523.

Di Vimercati, S. D. C., Foresti, S., Paraboschi, S. & Samarati, P. 2008. Access Control Models for Xml . *Handbook of Database Security, Applications and Trends.*

Di Vimercati, S. D. C., Marrara, S. & Samarati, P. 2005. An Access Control Model for Querying Xml Data. The 2005 Workshop on Secure Web Services, Fairfax, VA, USA, 1103029: ACM, 36-42.

Dietz, P. 1982. Maintaining Order in a Linked List. Proceedings of the fourteenth annual ACM symposium on Theory of computing USA: ACM, 122-127.

Duong, M. 2010. *Access Control Model and Labelling Scheme for Efficient Querying and Updating Xml Data.* PhD, Victoria University.

Duong, M. & Zhang, Y. 2005. Lsdx : A New Labeling Schema for Dynamically Updating Xml Data. 16th Australasian Database Conference, 185-193.

Duong, M. & Zhang, Y. 2008. An Integrated Access Control for Securely Querying and Updating Xml Data. The Nineteenth Conference on Australasian Database Gold Coast, Australia, 1378324: Australian Computer Society, Inc., 75-84.

Ekelhart, A., Fenz, S., Goluch, G., Steinkellner, M. & Weippl, M. 2008. Xml Security- Acomparative Literature Review. *The Journal of System and Software,* 81, 1715-1724.

Elmasri, R. & Navathe, S. 2003. *Fundamentals of Database Systems.* Addison Wesley.

Elmasri, R. & Navathe, S. 2007. *Fundamentals of Database Systems.* USA, Pearson International Edition.

Elmasri, R. & Navathe, S. 2011. *Database Systems Models, Languages, Design and Application Programming.* Boston, USA, PEARSON.

Eriksen, L. 2004. Xml Parsing with Sax and Dom - a Code Comparison. . The Australia Open Source Developers' Conference (OSDC2004), Melbourne.

Etoh, F., Takahashi, K., Hori, Y. & Sakurai, K. 2010. Study of Log File Dispersion Management Method. 10th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT), 371-374.

Farooqi, N. & North, S. 2011a. Developing a Dynamic Trust-Based Access Control Model for Xml Databases. University of Sheffield.

Farooqi, N. & North, S. 2011b. Trust-Based Access Control for Xml Databases. The 6th International Conference for Internet Technology and Secured Transactions (ICITST-2011), Abu Dhabi, UAE: IEEE Xplore, 764-765.

Farooqi, N. & North, S. 2012a. Evaluation of Access Process in Trust Based Access Control for Xml Databases. The 6th Saudi Scientific International Conference (SIC), Brunel University, London, UK.

Farooqi, N. & North, S. 2012b. Evaluation of Practical Trust Based Access Control for Xml Databases. The 7th International Conference for Internet Technology and Secured Transactions (ICITST), London, UK.: IEEE Xplore, 336-340.

Farooqi, N. & North, S. 2012c. Logging in Xml Databases: Xlog File for Trust Based Access Control. World Congress on Internet Security (WorldCIS-2012), Ontario, Canada IEEE Xplore, 174-175.

Farooqi, N. & North, S. 2012d. A Performance Evaluation of Logging in Xml Databases Using an Xlog File for Trust Based Access Control. *International Journal of Intelligent Computing Research (IJICR),* Volume 3, 337-341.

Farooqi, N. & North, S. 2013. Performance Evaluation of Trust Based Access Control for Xml Databases. *In- press , The Journal of Internet Technology and Secured Transactions (JITST),* 2, 1-8.

Feng, F., Lin, C., Peng, D. & Li, J. 2008. A Trust and Context Based Access Control Model for Distributed Systems. 10th IEEE International Conference on High Performance Computing and Communications, 629-634.

Feng, J.-B. & Yang, C.-C. 2010. A Mix of Role and Task-Based Access Control Model Research. Third International Conference on Information and Computing (ICIC), 66-69.

Fiebig, T., Helmer, S., Kanne, C., Moerkotte, G., Neumann, J. & Schiele, R. 2002. Anatomy of a Native Xml Base Managment System. *VLDB,* 11, 292-314.

Fiebig, T., Weber, H. & Harbarth. 2012. *Xml Database Managment System for an Xml Database Comprising Access - Protected Xml Data*. USA patent application 12/461,965.

Frank, T., Apel, M. & Schaeben, H. 2003. Web Integration of Gocad Using a 3d Xml Application Server In Proceedings of 23rd gOcad meeting CiteSeer, 1-10.

Gabillon, A. 2004. An Authorization Model for Xml Databases. The 2004 Workshop on Secure Web Service, Fairfax, Virginia, 1111351: ACM, 16-28.

Gabillon, A. 2005. A Formal Access Control Model for Xml Databases. Second VLDB Workshop on Secure Data Management (SDM): Springer, 86-103.

Gabillon, A. & Fansi, M. 2005. A Persistent Labelling Scheme for Xml and Tree Databases. Proceedings of the 1st International Conference on Signal-Image Technology and Internet-Based Systems, Yaounde, Cameroon 110-114.

Gire, F. & Idabal, H. 2008. Updates and Views Dependencies in Semi-Structured Databases. The 2008 international symposium on Database Engineering and Applications, Coimbra, Portugal, 159-168.

Gollmann, D. 2011. *Computer Security.* John Wiley & Sons.

Gonzalez, M., Perieto, M. & Nieto, M. 2009. A Study of Native Xml Databases. The 5th International Conference on Web Information Systems and Technologies (WEBIST), 89-92.

Green, T. J., Miklau, G., Onizuka, M. & Suciu, D. 2003. Processing Xml Streams with Deterministic Automata. LNCS, ed.The 9th International Conference Database Theory (ICDT), Italy, 173-189

Griffin, L., Butler, B., De Leastar, E., Jennings, B. & Botvich, D. 2012. On the Performance of Access Control Policy Evaluation. IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY), Chapel Hill, NC IEEE Xplore, 25 - 32

H. Molina , Ullman, J. & Widom, J. 2009. *Database Systems the Complete Book.* USA, Pearson International Edition.

Hada, S. & Kudo, M. 2000. *Xml Access Control Language:Provisional Authorization for Xml Documents* [Online]. Tokyo Research Laboratory, IBM Research. Available:

http://www.research.ibm.com/trl/projects/xml/xacl/xacl-spec.html [Accessed 20-12-2012].

Harold, E. 2005. *Managing Xml Data: Native Xml Databases Theory and Reality.* [Online]. IBM. Available: http://www.ibm.com/developerworks/xml/library/x-mxd4.html [Accessed 06-10-2011].

Harold, E. & Means, W. S. 2002. *Xml in a Nutshell.* O'Reilly.

He, H. & Yang, J. 2004. Multiresolution Indexing of Xml for Frequent Queries. The 20th International Conference on Data Engineering, USA, 683 - 694

Hégaret, P., Whitmer, R. & Wood, L. 2005. *Document Object Model (Dom)* [Online]. Available: http://www.w3.org/DOM/ [Accessed 17-03-2013].

Hitchens, M. & Varadharajan, V. 2001. Rbac for Xml Document Stores. *LNCS,* 2229, 131-143.

Hua, W. & Lili, S. 2010. Trust-Involved Access Control in Collaborative Open Social Networks. 4th International Conference on Network and System Security (NSS), 239-246.

Hui, F., Weinan, L., Wenchang, S., Zhaohui, L. & Bin, L. 2011. Trust-Oriented Access Control Based on Sources of Information Flow. The 13th International Conference on Advanced Communication Technology (ICACT), 797-801.

Izadi, K., Asadi, F. & Haghjoo, M. S. 2007. Xplc: A Novel Protocol for Concurrency Control in Xml Databases. IEEE/ACS International Conference on Computer Systems and Applications, 450-453.

Jeong, M. A., Kim, J.-J. & Won, Y. 2003. A Flexible Database Security System Using Multiple Access Control Policies. The Fourth International

Conference on Parallel and Distributed Computing, Applications and Technologies, 236-240.

Jia, L., Collins, M. & Nixon, P. 2009. Evaluating Trust-Based Access Control for Social Interaction. Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies 277-282.

Jin, X., Krishnan, R. & Sandhu, R. 2012. A Unified Attribute-Based Access Control Model Covering Dac, Mac and Rbac. The 26th Annual IFIP WG 11.3 Conference, DBSec 2012, Paris, France: Lecture Notes in Computer Science, 41-55.

Jittrawong, K. & Wong, R. K. 2007. Optimizing Xpath Queries on Streaming Xml Data. The Eighteenth Australasian Database Conference (ADC2007), Ballarat, Victoria, Australia, 73-82.

Jo, S.-M., Kim, Y.-K., Kouh, H.-J. & Yoo, W.-H. 2005. Access Control Model for Secure Xml Documents. Fourth Annual ACIS International Conference on Computer and Information Science, 352-357.

Jonge, A. 2008. *Comparing Xml Database Approaches* [Online]. IBM. Available: http://www.ibm.com/developerworks/library/x-comparexmldb/ [Accessed 31-10-2012].

Junbeom, H. & Dong Kun, N. 2011. Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems. *Parallel and Distributed Systems, IEEE Transactions on*, 22, 1214-1221.

Khaing, A. & Thein, N. 2006. A Persistent Labeling Scheme for Dynamic Ordered Xml Trees. IEEE/WIC/ACM International Conference on Web Intelligence, 498-501.

Kim, J., Park, S. & Seogpark. 2009. A New Labeling Scheme without Re-Labeling Using Circular Concepts for Dynamic Xml Data. The Ninth IEEE

International Conference on Computer and Information Technology., 318-323.

Kitagawa, N. & Yoshikawa, M. 2005. A Study on Efficient Access Control for Xml Documents. 21st International Conference on Data Engineering Workshops, 1230-1230.

Kozankiewicz, H., Leszczylowski, J. & Subieta, K. 2003. Updatable Xml Views. *Advances in Databases and Information Systems.* Springer Berlin Heidelberg.

Kuhn, D. R., Coyne, E. J. & Weil, T. R. 2010. Adding Attributes to Role-Based Access Control. *Computer,* 43, 79-81.

Landberg, A., Rahayu, J. & Pardede, E. 2010. Privacy-Aware Access Control in Xml Databases. 21st Australasian Database Conference (ADC), Brisbane, Australia: Australian Computer Society, 85-92.

Lang, J., Collins, M. & Nixon, P. 2009. Evaluating Trust-Based Access Control for Social Interaction. Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, 277-282.

Lawrence, R. 2004. The Space Efficiency of Xml. *Information and Software Technology,* 46, 753–759.

Lee, D. & Chu, W. 2000. Comparative Analysis of Six Xml Schema Languages. *ACM SIGMOD Record,* 29, 76-87.

Lee, D. & Yu, T. 2008. Xml Access Control. USA: The Pennsylvania State University.

Lee, K.-H., Whang, K.-Y., Han, W.-S. & Kim, M.-S. 2010. Structural Consistency: Enabling Xml Keyword Search to Eliminate Spurious Results Consistently. *The VLDB Journal,* 19, 503–529.

Li, F. & Hong, X. 2008. A Bitmap Indexing Scheme for Temporal Access Control to Xml Documents. IEEE International Symposium on IT in Medicine and Education 1062-1066.

Li, G., Feng, J., Wang, J. & Zhou, L. 2007. Effective Keyword Search for Valuable Lcas over Xml Documents. CIKM'07, Lisbon, Portugal, 1-10.

Li, L., Jiang, X. & Li, J. 2005. Enforce Mandatory Access Control Policy on Xml Documents. The 7th International Conference of Information and Communications Security (ICICS), Beijing, China: Lecture Notes in Computer Science, 336-349.

Li, Q. & Moon, B. 2001. Indexing and Querying Xml Data for Regular Path Expressions. The 27th Conference on Very Large Databases (VLDB), Roma, Italy, 361-370.

Li, Y. 2003. *The Xoo7 Benchmark* [Online]. Available: http://www.comp.nus.edu.sg/~ebh/XOO7.html [Accessed 18-03-2013].

Li, Y., Bressan, S., Dobbie, G., Lacroix, Z., Lee, M., Nambiar, U. & Wadhwa, B. 2001. Xoo7: Applying Oo7 Benchmark to Xml Query Processing Tool. ACM/CIKM, Atlanta, USA, 167-173.

Liefke, H. & Suciu, D. 2000. Xmill: An Efficient Compressor for Xml Data. The 2000 ACM SIGMOD International Conference on Management of Data New York, USA, 153-164.

Lin, A., Vullings, E. & Dalziel, J. 2006. A Trust-Based Access Control Model for Virtual Organizations. Fifth International Conference on Grid and Cooperative Computing Workshops, 557-564.

Lu, J., Ling, T. W., Chan, C.-Y. & Chen, T. 2005. From Region Encoding to Extended Dewey: On Efficient Processing of Xml Twig Pattern Matching. The 31st VLDB Conference, Trondheim, Norway, 193-204.

Ma, J., Logrippo, L., Adi, K. & Mankovski, S. 2010. Risk Analysis in Access Control Systems Based on Trust Theories. IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), 415-418.

Ma, X., Feng, Z., Xu, C. & Wang, J. 2008. A Trust-Based Access Control with Feedback. International Symposiums on Information Processing (ISIP), 510-514.

Merialdo., P. 1999. *Acm Sigmod Record: Xml Version* [Online]. Available: http://www.dia.uniroma3.it/Araneus/Sigmod/ [Accessed 18-03-2013].

Mlynkova, I. 2008. Xml Benchmarking:Limitations and Opportunities. Faculty of Mathematics and Physics, Department of Software Engineering, Charles University.

Mobley, P. 2011. *Learn from Your Mistakes* [Online]. Life Cycle Engineering. Available: http://www.lce.com/Learn_from_Your_Mistakes_458-item.html [Accessed 17-05-2013].

Mohan, S., Klinginsmith, J., Sengupta, A. & Yuqing, W. 2006a. Acxess - Access Control for Xml with Enhanced Security Specifications. Proceedings of the 22nd International Conference on Data Engineering, 171-171.

Mohan, S., Sengupta, A. & Wu, Y. 2005. Access Control for Xml: A Dynamic Query Rewriting Approach. The 14th ACM International Conference on Information and Knowledge Management, Bremen, Germany, 1099612: ACM, 251-252.

Mohan, S., Sengupta, A. & Wu, Y. 2006b. A Framework for Access Control for Xml. India: School of Informatics and Computing, Indiana University.

Mohan, S., Sengupta, A. & Wu, Y. 2007. A Rewrite Based Approach for Enforcing Access Constraints for Xml. *Lecture Notes in Computer Science (LNCS),* 4694, 1081-1089.

Molina, H., Ullman, J. & Widom, J. 2009. *Database Systems the Complete Book.* USA, Pearson International Edition.

Müldner, T., Fry, C., Miziołek, J. K. & Durno., S. 2009. Xsaqct: Xml Queryable Compressor. Balisage: The Markup Conference 2009, Montréal, Canada.

Murata, M., Tozawa, A., Kudo, M. & Hada, S. 2006. Xml Access Control Using Static Analysis. *ACM Transactions on Information and System Security (TISSEC),* 9, 292-324.

Myint, A. S. 2010. High Performance and Scalable Client-Based Access Control Model for Xml Databases. The 2nd International Conference on Computer and Automation Engineering (ICCAE), 369-372.

Na, N. & Guoqing, D. 2010. A New Labeling Scheme for Xml Trees Based on Mesh Partition. The 2nd International Conference on Future Computer and Communication (ICFCC), 353-356.

Nasa. 2001. *Gsfc Open Source Software* [Online]. Available: http://opensource.gsfc.nasa.gov/ [Accessed 18-03-2013].

Nazmul. 1999. *Should I Use Sax or Dom?* [Online]. Available: http://developerlife.com/tutorials/?p=28 [Accessed 18-03-2013].

Nicola, M., Gonzalez, A., Raghu, R., Zhang, Y., Kogan, I., Liu, M., Schiefer, B., Xie, G., Shum, P., Fichter, A. & Sommerlandt, M. 2007a. *Xml Database Benchmark: "Transaction Processing over Xml (Tpox)"* [Online]. Available: http://tpox.sourceforge.net/ [Accessed 25-03-2013].

Nicola, M., Kogan, I. & Schiefer, B. 2007b. An Xml Transaction Processing Benchmark In proceedings of the 2007 ACM SIGMOD international conference on Management of Data, Beijing, China, 937-948.

Noaman, A. & Almansour, A. 2012. Towards Achieving an Optimum Performance of Xml Data into Both Types of Xml Databases: Xml-Enabled Databases and Native Xml Databases. *Middle-East Journal of Scientific Research,* 2, 182-194.

O'neil, P., O'neil, E., Pal, S., Cseri, I., Schaller, G. & Westbury, N. 2004. Ordpaths: Insert-Friendly Xml Node Labels. Proceedings of the 2004 ACM SIGMOD international conference on Management of data, Paris, France, 1007686: ACM, 903-908.

Oasis. *Extensible Access Control Markup Language (Xacml)* [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml [Accessed 01-12-2012].

Oasis. *Oasis Security Services (Saml)* [Online]. OASIS. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security [Accessed 20-12-12].

Oasis. 2005. *Oasis Extensible Access Control Markup Language (Xacml)* [Online]. OASIS. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml [Accessed 20-12-12].

Onizuka, M. 2003. Lightweight Xpath Processing of Xml Stream with Deterministic Automata. The Twelfth International Conference on Information and Knowledge Management (CIKM), New Orleans, Louisiana, USA, 342-349.

Oqbuji, U. 2004a. *A Hands-on Introduction to Schematron* [Online]. IBM. Available: http://www.ibm.com/developerworks/xml/tutorials/x-schematron/section2.html [Accessed 05-11-2012].

Oqbuji, U. 2004b. *A Survey of Xml Standards: Part 1* [Online]. IBM. Available: http://www.ibm.com/developerworks/xml/library/x-stand1/index.html [Accessed 31-10-2012].

Palani, G. 2011. *Investigate Current Xml Tools* [Online]. IBM. Available: http://www.ibm.com/developerworks/xml/library/x-xmltools/index.html [Accessed 31-10-2012].

Papamarkos, G., Zamboulis, L. & Poulovassilis, A. 2009. Xml Databases. London, UK: School of Computer Science and Information Systems.

Park, J. S., Costello, K. P., Neven, T. M. & Diosomito, J. A. 2004. A Composite Rbac Approach for Large, Complex Organizations. The Ninth ACM Symposium on Access Aontrol Models and Technologies, Yorktown Heights, New York, USA, 990063: ACM, 163-172.

Park, J. S. & Giordano, J. 2006. Role-Based Profile Analysis for Scalable and Accurate Insider-Anomaly Detection. 25th IEEE International Performance, Computing, and Communications Conference, 70-74.

Qi, N., Kudo, M., Myllymaki, J. & Pirahesh, H. 2005. A Function-Based Access Control Model for Xml Databases. The 14th ACM International Conference on Information and Knowledge Management, Bremen, Germany, 1099577: ACM, 115-122.

Radiya, A. & Dixit, V. 2000. *The Basics of Using Xml Schema to Define Elements* [Online]. IBM. Available: http://www.ibm.com/developerworks/xml/library/xml-schema/index.html [Accessed 05-11-2012].

Rafiei, D., Moise, D. L. & Sun, D. 2006. Finding Syntactic Similarities between Xml Documents. the 17th International Conference on Database and Expert Systems Applications, USA, 512-516

Rashid, Z., Basit, A. & Anwar, Z. 2010. Trdbac: Temporal Reflective Database Access Control. The 6th International Conference on Emerging Technologies (ICET), 337-342.

Ray, E. 2003. *Learning Xml.* O'Reilly.

Roantree, M., Noonan, C. & Murphy, J. 2007. Specifying and Optimising Xml Views. The 24th British national conference on Databases, Glasgow, UK, 138-146.

Runapongsa, K., Patel, M., Jagadish, H., Chen, Y. & S., A.-K. 2006a. *The Michigan Benchmark* [Online]. Available: http://www.eecs.umich.edu/db/mbench/description.html [Accessed 18-03-2013].

Runapongsa, K., Patel, M., Jagadish, H., Chen, Y. & S., A.-K. 2006b. The Michigan Benchmark: Towards Xml Query Performance Diagnostics. *International Journal of Information systems,* 31, 73-97.

Ryutov, T., Zhou, L., Neuman, C., Leithead, T. & Seamons, K. E. 2005. Adaptive Trust Negotiation and Access Control. The Tenth ACM Symposium on Access Control Models and Technologies, Stockholm, Sweden, 1064004: ACM, 139-146.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L. & Youman, C. E. 1996. Role-Based Access Control Models. *Computer,* 29, 38-47.

Sandhu, R. S. & Samarati, P. 1994. Access Control: Principle and Practice. *Communications Magazine, IEEE,* 32, 40-48.

Sans, V. & Laurent, D. 2008. Prefix Based Numbering Schemes for Xml: Techniques, Applications and Performances. *Proc. VLDB Endow.*, 1, 1564-1573.

Sax. 2004. *About Sax* [Online]. Available: http://sax.sourceforge.net/ [Accessed 18-03-2013].

Schmidt, A. 2003. *Xmark — an Xml Benchmark Project* [Online]. Available: http://www.xml-benchmark.org/ [Accessed 18-03-2013].

Schmidt, A., Waas, F., Kersten, M., Carey, D., Manolescu, I. & R, B. 2002. Xmark: A Benchmark for Xml Data Management. International Conference on Very Large Databases, Hong Kong, China, 1-12.

Schmidt, A., Waas, F., Kersten, M., Florescu, D., Carey, M., Manolescu, J. & Busse, R. 2001. Why and How to Benchmark Xml Databases. *ACM/SIGMOD Record,* 30, 27-32.

Shen, H.-B. 2010. A Semantic- and Attribute-Based Framework for Web Services Access Control. The 2nd International Workshop on Intelligent Systems and Applications (ISA), 1-4.

Silvasti, P., Sippu, S. & Soininen, E. 2009. Schema-Conscious Filtering of Xml Documents. EDBT Saint Petersburg, Russia, 970-981.

Singh, S. 2011. Trust Based Authorization Framework for Grid Services. *Journal of Emerging Trends in Computing and Information Sciences,* 2, 136-144.

Staken, K. 2001. *Introduction to Native Xml Databases* [Online]. O'Reilly. Available: http://www.xml.com/pub/a/2001/10/31/nativexmldb.html [Accessed 06-10-2011].

Steedman, M., Osborne, M., Sarkar, A., Clark, S., Hwa, R., Hockenmaier, J., Ruhlent, P., Bakeri, S. & Crimt, J. 2003. Bootstrapping Statistical Parsers

from Small Datasets. The Tenth Conference on European Chapter of the Association for Computational Linguistics Stroudsburg, PA, USA 331-338.

Steegmans, B., Bourret, R., Cline, O., Guyennet, O., Kulkarni, S., Priestley, S., Sylenko, V. & Wahli, U. 2004. *Xml for Db2 Information Integration.* USA, IBM.

Suciu, D. 2002. *Xml Data Repository* [Online]. University of Washington. Available: http://www.cs.washington.edu/research/xmldatasets/ [Accessed 18-03-2013].

Sun, L. & Wang, H. 2011. A Purpose-Based Access Control in Native Xml Databases. *Concurrncy and Computation:Practice and Experience,* 24, 1154–1166.

Sun, L., Wang, H., Jururajin, R. & Sriprakash, S. 2010. A Purpose Based Access Control in Xml Databases System. The 4th International Conference on Network and System Security (NSS), 486-493.

Teorey, T., Lightstone, S. & Nadeau, T. 2006. *Database Modeling and Design.* USA, Morgan Kaufmann.

Thimma, M., Tsui, T. K. & Luo, B. 2013. Hyxac: A Hybrid Approach for Xml Access Control. The 18th ACM Symposium on Access Control Models and Technologies Amsterdam, Netherlands: ACM, 113-123.

Tidwell, D. 2002. *Introduction to Xml* [Online]. IBM. Available: http://www.ibm.com/developerworks/xml/tutorials/xmlintro/section2.html [Accessed 31-10-2012].

Tizag. *Xml Comments* [Online]. Available: http://www.tizag.com/xmlTutorial/xmlcomment.php [Accessed 05-10-2011].

Totalxml. 2011. *History of Xml* [Online]. Available: http://www.totalxml.net/history-xml.php [Accessed].

Tran, H., Hitchens, M., Varadharajan, V. & Watters, P. 2005. A Trust Based Access Control Framework for P2p File-Sharing Systems. The 38th Annual Hawaii International Conference on System Sciences, 302c-302c.

Treebank. 1999. *The Penn Treebank Project* [Online]. Available: http://www.cis.upenn.edu/~treebank/ [Accessed 18-03-2013].

Trochim, W. 2006. *Reliability* [Online]. Available: http://www.socialresearchmethods.net/kb/reliable.php [Accessed 17-05-2013].

Vakali, A., Catania, B., Maddalena, A. & Aristotle Univ., G. 2005. Xml Data Stores: Emerging Practices *IEEE Internet Computing*, 9, 62-69.

Vela, B., Mazon, J., Blanco, C., Medina, E., Trujillo, J. & Marcos, E. 2013. Development of Secure Xml Data Warehouses with Qvt. *In-press: Information and Software Technology*.

Verma, B., Kumar, S. & Sharma, P. 2012. A Novel Approach for Multi-Tier Security for Xml Based Documents. *IOSR Journal of Computer Engineering (IOSRJCE),* Volume 5, 1-4.

W1. 2009. *Sax Vs Dom. How to Choose between Dom and Sax?* [Online]. Available: http://geekexplains.blogspot.co.uk/2009/04/sax-vs-dom-differences-between-dom-and.html [Accessed 17-03-2013].

W2. 2008. *Parsers? Dom Vs Sax Parser* [Online]. Available: http://dev.fyicenter.com/Interview-Questions/Java-1/Parsers_DOM_vs_SAX_parser.html [Accessed 17-03-2013].

W3c. 2003. *Document Object Model Faq* [Online]. Available: http://www.w3.org/DOM/faq.html#whybother [Accessed 17-03-2013].

W3c. 2010. *What Is Xml?* [Online]. Available: http://www.w3.org/standards/xml/core [Accessed 29-9-2011].

W3schools. 2013a. *Introduction to Xml.* [Online]. Available: http://www.w3schools.com/xml/xml_whatis.asp [Accessed 29-09-2011].

W3schools. 2013b. *Xquery Tutorial* [Online]. Available: http://www.w3schools.com/xquery/default.asp [Accessed 15-05-2013].

Waldt, D. 2010. *Six Strategies for Extending Xml Schemas in a Single Namespace* [Online]. IBM. Available: http://www.ibm.com/developerworks/xml/library/x-xtendschema/index.html [Accessed 05-11-2012].

Walsh, N. 1998. *A Technical Introduction to Xml* [Online]. O'Rielly. Available: http://www.xml.com/pub/a/98/10/guide0.html?page=3 [Accessed 05-10-2011].

Wang, H., Park, S., Fan, W. & Yu, P. 2003. Vist: A Dynamic Index Method for Querying Xml Data by Tree Structure. The 2003 ACM SIGMOD International Conference on Management of Data, California, USA, 110-121.

Wang, J. & Osborn, S. L. 2004. A Role-Based Approach to Access Control for Xml Databases. The Ninth ACM Symposium on Access Control Models and Technologies, Yorktown Heights, New York, USA, 990047: ACM, 70-77.

Whatley, K. 2009. *Xml Basics for New Users* [Online]. IBM. Available: http://www.ibm.com/developerworks/library/x-newxml/ [Accessed 31-10-2012].

Wong, R. K., Lam, F. & Shui, W. M. 2007. Querying and Maintaining a Compact Xml Storage. The International World Wide Web Conference Committee(IW3C2), Banff, Alberta, Canada, 1073-1082.

Wu, X., Lee, M. L. & Hsu, W. 2004. A Prime Number Labelling Scheme for Dynamic Ordered Xml Trees. 20th International Conference on Data Engineering, 66-78.

Xing, H.-F., Cui, B.-L. & Xu, L.-L. 2010. A Mixed Access Control Method Based on Trust and Role. The Second IITA International Conference on Geoscience and Remote Sensing 552-555.

Xu, L., Ling, T. & Wu, H. 2010. Labeling Dynamic Xml Documents:An Order-Centric Approach. *Knowledge and Data Engineering, IEEE Transactions on*, PP, 1-1.

Xu, L., Ling, T. W., Wu, H. & Bao, Z. 2009. Dde: From Dewey to a Fully Dynamic Xml Labeling Scheme. Proceedings of the 35th SIGMOD international conference on Management of data, Providence, Rhode Island, USA, 1559921: ACM, 719-730.

Xu, Y. & Papakonstantinou , Y. 2005. Efficient Keyword Search for Smallest Lcas in Xml Databases. In proceedings of the 2005 ACM SIGMOD international conference on Management of Data, Baltimore,Maryland,USA, 527-538.

Yang, X., Shen, J. & Si, Z. 2010. A New Trust Degree-Based Access Control Method for Semantic Web Services. International Conference on Intelligent Computation Technology and Automation (ICICTA), 250-253.

Yao, B., Özsu, M. & Khandelwal, N. 2004. Xbench Benchmark and Performance Testing of Xml Dbmss. The International Conference of Data Engineering, 621-632.

Yi, H. & Brajendra, P. 2003. Identification of Malicious Transaction in Database Systems. The Seventh International Database Engineering and Applications Symposium, China: IEEE Computer Society, 329-335.

Yi, H. & Panda, B. 2003. Identification of Malicious Transactions in Database Systems. Seventh International on Database Engineering and Applications Symposium, 329-335.

Yu, T., Srivastava, D., Lakshmanan, L. V. S. & Jagadish, H. V. 2002. Compressed Accessibility Map: Efficient Access Control for Xml. The 28th International Conference on Very Large Databases, Hong Kong, China, 1287411: VLDB Endowment, 478-489.

Yu, T., Srivastava, D., Lakshmanan, L. V. S. & Jagadish, H. V. 2004. A Compressed Accessibility Map for Xml. *ACM Trans. Database Syst.*, 29, 363-402.

Yuan, E. & Tong, J. 2005. Attributed Based Access Control (Abac) for Web Services IEEE International Conference on Web Services, Florida,USA: IEEE Computer Society, 561-569.

Yuanbo, Q., Xiaoguang, H. & Ji, F. 2009. An Approach to Construct Secure View for Xml. International Conference on Management and Service Science MASS '09. , 1-4.

Zhang, D.-Z. & Xue, Y.-S. 2005. An Extended Mandatory Access Control Model for Xml. *Advances in Computer Science -Asian 2005 Data Management on the Web.* Springer Berlin Heidelberg.

Zhang, L. & Rao, R. 2010. Trust Based Access Control Framework for R-Osgi. The 2nd International Workshop on Database Technology and Applications (DBTA) 1-5.

Zhang, N., Haas, P. J., Josifovski, V., Lohman, G. M. & Zhang, C. 2005. Statistical Learning Techniques for Costing Xml Queries. The 31st VLDB Conference, Trondheim, Norway, 289-300.

Zhang, P. & Dong, G. 2010. A New Labeling Scheme Using Vectors Based on Polar Coordinate System for Dynamic Xml Data. Second Pacific-Asia Conference on Circuits,Communications and System (PACCS), 167-170.

Zhang, S. 2006. *The Design and Implementation of Australia's Virtual Herbarium System.* MSc Dissertation, The University of Adelaide.

Zhang, X., Park, J. & Sandhu, R. 2004. Schema Based Xml Security: Rbac Approach. *Data and Applications Security* Springer US.

Zhang, Y., Wu, M., Wu, L. & Li, Y. 2013. Attribute-Based Access Control Security Model in Service-Oriented Computing. The International Conference on Cybernetics and Informatics, China: Lecture Notes in Electrical Engineering, 1473-1479.

Zhao, L., Liu, S., Li, J. & Xu, H. 2010. A Dynamic Access Control Model Based on Trust. International Conference on Environmental Science and Information Application Technology (ESIAT), 548-551.

Zhu, H., Jin, R. & Lu, K. 2007. A Flexible Mandatory Access Control Policy for Xml Databases. The 2nd International Conference on Scalable Information Systems, Suzhou, China, 1366890: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 1-4.

Zhu, H., Lu, K. & Jin, R. 2009. A Practical Mandatory Access Control Model for Xml Databases. *Information Sciences,* 179, 1116-1133.

14 APPENDIX I: FULL RESULTS FOR THE ACCESS CONTROL MODUL EXPERIMENT

14.1 Full Results in Platform One

- **SIGMOD Record**

Number of users	Node searched time	User searched time	TV searched time	Total time for the whole code
50	37.1 (26.2%)	4.34 (3.1%)	1.42 (1%)	141
100	37.7 (25.7%)	7.52 (5.1%)	1.41 (0.9%)	146
1000	35.6 (20.3%)	19.1 (10.9%)	1.44 (0.8%)	175

- **NASA**

Number of users	Node searched time	User searched time	TV searched time	Total time for the whole code
50	724(37.2%)	4.5 (0.2%)	1.96 (0.1%)	1947
100	729 (37.3%)	7.49 (0.4%)	1.95 (0.1%)	1955
1000	724(36.4%)	20.3 (1%)	1.97 (0.1%)	1993

- **Treebank**

Number of users	Node searched time	User searched time	TV searched time	Total time for the whole code
50	1990 (35.3%)	4.24 (0.1%)	1.45 (0%)	5636
100	1990 (35.2%)	7.33 (0.1%)	1.54 (0%)	5656
1000	1995 (35.2%)	20.8 (0.4%)	1.50 (0%)	5670

14.2 Full Results in Platform Two

- **SIGMOD Record**

Number of users	Node searched time	User searched time	TV searched time	Total time for the whole code
50	45.6 (32.3%)	3.36 (2.4%)	0.99 (0.7%)	141
100	44.8 (31.4%)	6.20 (4.3%)	1.0 (0.7%)	146
1000	30.9 (17.9%)	18.7 (10.8%)	0.98 (0.8%)	172

- **NASA**

Number of users	Node searched time	User searched time	TV searched time	Total time for the whole code
50	760(35.9%)	3.29 (0.2%)	1.52 (0.1%)	2115
100	767 (35.9%)	5.97 (0.3%)	1.47 (0.1%)	2137
1000	762 (35.4%)	18.1 (0.8%)	1.44 (0.1%)	2151

- **Treebank**

Number of users	Node searched time	User searched time	TV searched time	Total time for the whole code
50	3967 (15.4%)	3.44 (0.0%)	1.67 (0%)	25742
100	4017 (15.6%)	6.11 (0.0%)	1.42 (0%)	25818
1000	3983 (15.4%)	18.2 (0.1%)	1.25 (0%)	25908

15 APPENDIX II: EXTRA RESULTS FOR THE TRUST MAINTENANCE EXPERIMENT

15.1 Results in Platform Two

	The Number of Users		
	50	**100**	**1000**
EF&BTF = 0	109	122	188
EF&BTF = 0.25	111	125	190
EF&BTF = 0.5	112	127	191
EF&BTF = 0.75	113	127	195
EF&BTF = 1	115	129	197

16 APPENDIX III: EXTRA RESULTS FOR THE ACCESS SUPERVISION EXPERIMENT

16.1 Results in Platform Two

- The Time consumed for the first query Q1: Deleting/RootNode.

Number of users	SIGMOD	NASA	Treebank	Average time (seconds)
50	86.8	86.1	87.4	0.08
100	95.5	95.5	94.2	0.09
1000	126	129	131	0.13

- The Time consumed for the second query Q2: //NodeName.

Number of users	SIGMOD	NASA	Treebank
50	243	2228	26282
100	253	2234	26311
1000	281	2261	26361