XML DOCUMENTS AND ACCESS CONTROL MANAGEMENT

LiLi Sun

BSc(Sci), MSc(Sci)

A Dissertation submitted for the degree of Doctor of Philosophy

The Department of Mathematics and Computing

The University of Southern Queensland

January 2010

Statement

I hereby declare that the work presented in this dissertation is my own and is, to the best of my knowledge and belief, original except as acknowledged in the text. It has not previously been submitted either in whole or in part for a degree at this or any other university.

Lili Sun

Signature of Candidate Date

ENDORSEMENT

Signature of Supervisor Date

Acknowledgement

I express my sincere gratitude and appreciation to many people who made this PhD thesis possible. I would like to thank my PhD supervisor Dr Yan Li of the University of Southern Queensland (USQ) for her general view on research, invaluable advice and encouragements during the realization of this research.

I sincerely thank the Department of Mathematics and Computing, Faculty of Sciences and Research and Higher Degrees office of USQ for providing the excellent study environment and the financial support. It is a great pleasure to study at the Department of Mathematics and Computing.

I also acknowledge all the help from those who carefully read the dissertation and made the English corrections.

Finally, my appreciation goes to my parents Lianshu Sun, Yuntao Cai, my husband Hua and daughter Nana for their love and affection. I could not be able to complete my PhD study without their encouragement and support.

Abstract

Security requirements of confidentiality, integrity and availability are essential for business online. With the growing acceptance of Extensible Markup Language (*XML*) technologies for documents and protocols, it is necessary that security should be integrated with *XML* solutions. *XML* was designed to transport and store data, especially over the Internet. Information exchanges on the Internet should meet precise protection requirements: fine-grained authenticity, secrecy, non-repudiation and access control. The requirements have to address for *XML* documents and *XML* applications [15]. *XML* has become an important universal language for the Internet-based business world [24]. An *XML* document can be generated from various resources with varying security requirements, such as Authentication, Authorization, Integrity, Signature, Confidentiality, Privacy and Digital Rights Management. According to these requirements, the main relevant developments of *XML* Security standards are [12, 15, 37]:

- *XML* Digital Signature,
- *XML* Encryption,
- *XML* Key Management,
- Security Assertion Markup Language (SAML),
- *XML* Access Control Markup Language (XACL).

XML is a fundamental component in many *XML* web services and it is used to store and exchange data in the Internet environment that may include private messages. It overcomes the complexity of Standard Generalized Markup Language (*SGML*) and the user can define document structures, removing the limit of the fixed tags in Hypertext Markup Language (*HTML*) [26]. *XML* Security therefore must be integrated with *XML* in such a way as to maintain the advantages and capabilities of *XML* while adding necessary security capabilities.

Traditional access control models for *XML* primarily consider static authorization decisions based on the subjects' permissions on target objects [91]. The models have been used only on the control of access to server-side objects and static authorization decisions are not assessed once an access permission is granted. Static authorizations for *XML* documents are performed without ongoing evaluating of query expressions against an actual database application. For example, a prepaid mobile needs ongoing checking to determine whether or not a call can continue or will be denied. To cope with these problems, recently proposed usage access control is a new access control model, which extended traditional access control models in multiple aspects: dynamic authentication, pre-authorization, ongoing-authorization, obligation and conditions [27].

In this dissertation, we aim to provide a bridge between the existing security technologies and the secure methods for *XML* documents. We extend existing web security technologies for *XML* documents. Usage access control models are analysed for *XML* schema and Document Type Definition (*DTD*) level authorizations. As validation of *XML* documents, technologies for *XML* databases have been enhanced and improved through usage access management. The theory developed in this dissertation can be applied in electronic services, such as E-learning. Role-based access

control (*RBAC*) is an access approach and permission-role assignments are main parts in *RBAC*. The new authorization algorithms for permission-role assignments in *RBAC* have been developed.

Publications Based on this Thesis

1. L. Sun and Y. Li, Using Usage control to access *XML* Databases, International Journal of Information Systems in the Service Sector, Vol. 1, No. 3, pages 32-44, July-September 2009.

2. L. Sun and Y. Li, *XML* and Web Service Security, The 12th International Conference on Computer Supported Cooperative Work in Design, Vol. 4823/2008, pages 765-770, IEEE, April 2008.

3. L. Sun and Y. Li, *XML* Schema in *XML* Documents with Usage control, International Journal of Science & Network Security, Vol. 6, pages 170-177, December 2007.

4. L. Sun, H. Wang and Y. Li, Protecting disseminative information in E-Learning, Advances in Web Based Learning -ICWL 2007, Vol. 4823/2008, pages 554-564, Springer, August 2007.

5. L. Sun and Y. Li, DTD Level Authorization in *XML* Documents with Usage control, International Journal of Science & Network Security, Vol. 6, pages 244-250, November 2006.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents the motivation and objectives of this dissertation. There are three sections in the chapter. In the first section, technology issues with their unsolved problems are introduced in *XML* and access control management. The issues are *XML* and Web service security, *XML* access control models and usage access control. The major contributions are presented in the second section. The objectives and the organization of this dissertation are demonstrated in the third and fourth section, respectively.

1.1 Overview and motivation

Information is currently accessible anywhere, at anytime. It is revolutionizing the way we work. Its impact has already improved our daily life and has brought many benefits for people who can share and distribute information all over the world. With regards to the digital content industry, this allows for the consideration of alternatives to tradition modes of distribution of data and provides many new data models such as semi-structured data that are applied. Therefore, information can be made widely available to customers.

The Web is becoming the main information disseminator for private and public organizations. A vast amount of information is exchanged every day ranging from simple to complex files from Web services. In some cases, the availability of the information has to be restricted to access only for registered users and failing to do so may cause big economic losses [13]. Therefore, confidential information is becoming important. For example in the case of student folders, their information can be shared only among administrators and their supervisors. A leakage of students' private information may cause trouble for a university.

1.1.1 *XML* and web services security

Meeting security requirements for privacy, confidentiality and integrity is essential in order to move business online. Security has always been important in the business world to make sure information is used appropriately. A complete solution to data security must meet the following three requirements: 1) *secrecy* or *confidentiality* refers to protection of data. 2) *integrity* refers to the prevention of unauthorized and improper data modification. 3) *availability* refers to prevention and recovery from hardware and software errors and data access denials [13, 43]. However, in today's web-based business environment, using physical security no longer works as well as it did in the past since there is too much to administrate, too many applications, too many variations, and too rapid a pace of technology changes to design a single infrastructure to meet all requirements effectively [13]. Extensible security standards are required to fit with new technologies to enable open distributed systems, application integration and content management.

Over the past several years, there has been a tremendous surge of interest in *XML* as a universal, queryable representation for data. *XML* promoted by the World Wide

Web Consortium (*W3C*) is rapidly emerging as a new standard language for semi-structured data representation and exchange over the Internet. *XML* web service is a platform-independent Web application that accepts requests from different systems on the Internet. *XML* languages is text based and designed to be extended and combined. It should naturally provide integrity, confidentiality and availabilty benefits to *XML* documents or portions of these documents in a way that does not prevent further processing by standard *XML* tools [4]. Existing security technologies provide a set of core security algorithms and technologies that can be used in *XML* security, but the actual formats used to implement security requirements are inappropriate for most *XML* security applications [3]. These technologies use binary formats with specialized software for interpretation and use. These standards are neither designed for use with *XML* documents nor do they support common *XML* technical approaches, such as specifying content with uniform resource identifier strings (*URIs*). Furthermore, some existing security technologies assume that security-specific software is integrated with applications to enable security, but this is not always the case in practice due to the details of custom integration.

Techniques to satisfy these requirements are: Authentication, Access control and Encryption. It is clear that *XML* is a powerful development technology, but it may have security problems that are related to the sensitive data in *XML* documents. For example, an *XML* document may be generated from various resources with varying security requests due to its ability to express the complex relationship between data. Alternatively, a user may like to limit access to particular parts of an *XML* document with signatures. In order to ensure the integrity of content and transactions, to maintain privacy and confidentiality, and to make sure that information is used appropriately, security is becoming more and more important for *XML* documents

and applications [12]. Several approaches have been designed for the security of *XML* documents, such as encryption and decryption, Secure Sockets Layer (*SSL*) and Simple Object Access Protocol (*SOAP*) [43]. But these approaches have limitations. For instance, encryption and decryption skills focus on the protection at the file level, but not on a systematic level [53]. They are used in protection of communications between servers and clients rather than dissemination from clients. *SSL* [54], the recent techniques used in Intranet assume that all computers know *XML* web services. *SSL* is applied to encrypt and decrypt messages which are exchanged between clients and services. It can protect messages from unauthorized reading while these messages are in transition and also verify that incoming messages actually come from the correct sender. However, *SSL* is not satisfactory in Web service environments, specially in association with *XML* messages used in Simple Object Access Protocol (*SOAP*) based communications [5]. *XML* signatures are designed for the security of *XML* document transactions. They are involved in authentication, data integrity and support for non-repudiation to the data that they sign. They ensure that signatures cannot be verified without interaction with the signer. An *XML* undeniable signature [114] method builds a bridge between the existing *XML* technologies and *RSA* algorithm based undeniable signature technologies [114].

Security technologies provide security algorithms and technologies that can be used in *XML* security, but for many of them the actual formats used to implement security requirements are inappropriate for most applications. Usually these standards are not designed for use with *XML* and do not support common *XML* technical approaches for managing content, such as specifying content with Uniform Resource Identifier strings (URIs) or using other *XML* standard definitions for locating portions of *XML* content (like *XML* Path Language[*XPath*]) [100]. At the

same time these standards use binary formats that require specialized software for interpretation and use, even for extracting portions of the security information. In addition, some current existing security technologies, such as Secure Socket Layer (*SSL*) [54], Transport Layer Security (*TLS*) and Hypertext Transfer Protocol Secure *HTTPS* [100] provide several specifications for web services to enable security. But there are some issues with these schemes. *SSL* and *TLS* both provide transport level security, not message level security. They are point-to-point security only and can not handle end-to-end multi-hopped messaging security. Security only when data is in transition, does not secure data off transition. *HTTPS* does not support non-repudiation.

In 2002, several specifications were proposed for securing *XML*-based applications and web services. These standards support the integration of security functionalities into their *XML* based applications.

Figure 1.1 shows some of the most important specifications for *XML* and Web service security. This provides a standard framework for *XML* based applications. These specifications mainly include the following aspects:

- *XML* Digital Signature
- *XML* Encryption
- *XML* Key Management Specification
- Security Assertion Markup Language
- *XML* Access Control Markup Language
- WS-Security

Usually Simple Object Access Protocol (SOAP) [22] is used for message transport. *XML* digital signature and *XML* encryption are used for data confidentiality and integrity. Security Assertion Markup Language (SAML) [7] focuses on authentication assertions. *XML* Access Control Markup Language (XACML) [6] is for information access control. *XML* Key Management Specification (XKMS) [2] is used to manage Public key infrastructure and Web service security brings standards together [92].

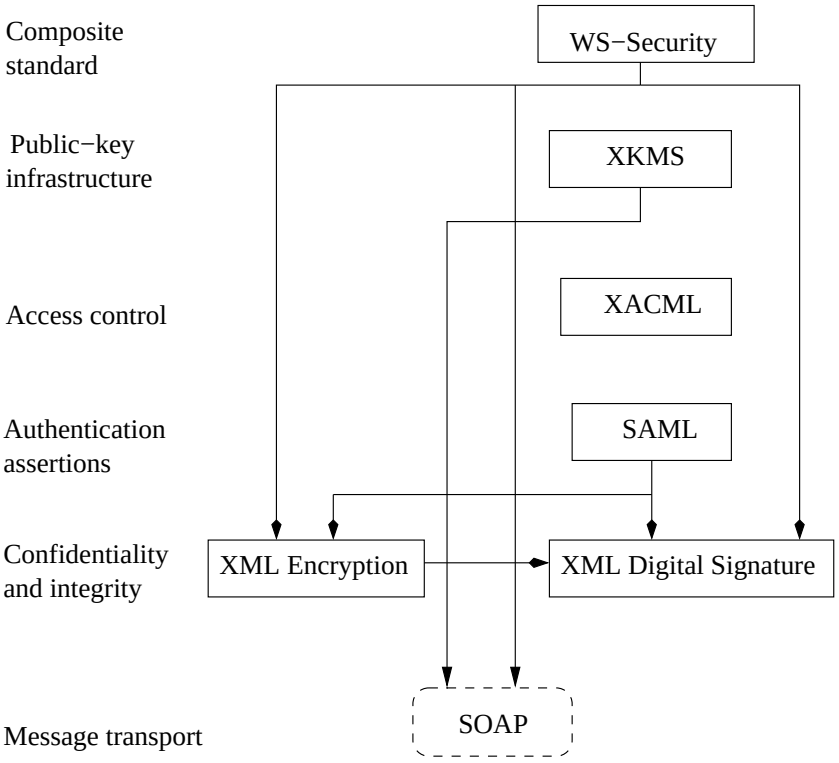

Figure 1.1: XML security standards

1.1.2 *XML* access control models and usage access control

There are now fields of *XML* security technology such as *XML* encryption, *XML* signature, XKMS, and XACML [120]. Thus far, the existing studies and products for security of *XML* documents, transmission-layer security protocols including *XML* signature and encryption, and access control to information in web environments,

were mostly related to *HTML* documents. This was file-based access control and failed to deal with access control by means of partial information, or the principal advantage of *XML*. So an access control method with the advantages of *XML* techniques become necessary [43].

The existing web-based access control models can describe authorization in a unit or part of files. However, these methods fail to create access based on the meaning of information. In order to deal with information by its meaning, which characterizes *XML* documents most remarkably, access to such small units as elements [19] is required. To reduce administration cost and complexity and to improve the security of management, in this dissertation we analyze how to use usage access control to manage *XML* documents. Recently, usage access control has been widely used in database system and operating system products. The usage access control is a unified approach to capture a number of extensions for the traditional access control. An access control decision is based on the subject and/or the object attributes, which can be changed as the side-effects of using the access right, resulting in possible changes to future access control decisions. Hence the safety question in usage access control is all the more pressing since every access can potentially enable additional permissions due to the mutability of attributes in usage access control.

There has been little research done on the use of usage access control in web database systems. Methods of the use of usage access control to manage *XML* database system is another challenge in this project. We have analyzed usage access control and its management for *XML* database systems and E-Learning in Chapter 6 and Chapter 8, in this dissertation.

1.2 Contributions

This thesis has made the following major contributions for protecting *XML* documents:

1. A connection between *XML* documents and usage access control has been built after reviewing traditional access methods and the structure of *XML*. Two main validation specification mechanisms *DTD* and *XML* Schema are discussed. The features of the latest usage access control are discussed including obligations, conditions and continuity properties. The motivations of adopting usage access control for *XML* documents are analysed.

2. *XML* was designed to transport and store data which is related to web services. The security standards for *XML* documents and web services are different. We discuss the differences and present the standard specifications for both *XML* documents and web service. Each standard for protecting *XML* based documents and relations between these standards are discussed. Usage control has been considered as the next generation access control model with distinguishing properties of decision continuity, we present a usage control model to protect information distributed on the web, which allows the access restrictions directly on structures and documents. The model not only supports complex constraints for *XML* components, such as elements, attributes and datatypes but also provides mechanisms to build rich reuse relationships between models and databases.

3. Authorization models with usage access control to manage access both at the instance-level and schema-level are developed. Traditional access control is based on an access request, an algorithm that computes a view of an *XML* document target based on the user's requirements rights. An authorization decision for a subject

access request to target resources is made before access. The authorization permission is not assessed during the accessing period even though access conditions are changed. Authorization decisions in usage access control are checked and made before access, but also repeatly checked during the access period. It may revoke the access permission according to the changes of the subject or object attributes. Furthermore, obligation and conditions are decision factors for the access management of *XML* documents.

4. We present a usage control model to protect information distributed on the web, which allows the access restrictions directly at schema-level and *XML* document-level. *XML* Schema provides a mechanism to build relationships between schemas and elements which are important for securing *XML* documents. Authorization models with usage access control for *XML* Schema are designed. It includes pre-Authorizations, ongoing-Authorizations, pre-Obligations, ongoing-Obligations, pre-Conditions and ongoing-Conditions models.

5. We provide a fundamental analysis for a usage control model on *XML* databases, especially the Object-oriented database systems (*OODB*) authorization model. The *OODB* authorization model is used for database access management, but it has only one authorization component: object. It doesn't include obligation and condition components. Usage control encompasses traditional access control, trust management and it provides an approach for the next generation of access control. It covers both security and privacy issues of current business and information systems. Compared to the OODB authorization model, the usage control model for accessing *XML* databases has a wide scope of applications.

6. Role-based access control (*RBAC*) is a famous access approach and permission-role assignments are one major part in *RBAC*. We have developed new authorization

algorithms for permission-role assignments that are based on relational algebra operations. They are the authorization granting algorithm, weak revocation algorithm, and strong revocation algorithm. The algorithms can automatically check conflicts when granting more than one permission to a role in a system. They can prevent users associated with roles from accessing unauthorized use of facilities when the permissions of the roles are changed within the organization and demand the modification of security rights. The permissions can be allocated without compromising the security in *RBAC* and provide secure management for systems.

7. We analyse the dissemination of payment-free-type digital information in E-learning, and discuss access models and architectures by using usage control. Different types of access models are built for digital documents. To protect digital documents from malicious dissemination, we have analysed reference monitors on both server and client sides and obtained several secure architecture solutions.

1.3 Objectives of the dissertation

XML is widely used today in a large variety of applications and industry products, as it has become the most important standard for describing data and documents circulated across the web. It is becoming a vital component in the emerging electronic business infrastructure. We need trustworthy, secure *XML* messages to form the basis of business transactions. Early research work about *XML* was not directly related to access control and security, because *XML* was initially introduced as a data format for documents. Therefore, many researchers assumed well-known techniques for securing documents to be straightforwardly applicable to *XML* data. But the way *XML* has been positioned has caused some to question whether additional measures will be necessary. For example, some users will be able to control their

interaction with Web sites by pulling the information they are interested in from dynamically generated *XML* documents. However, different users may well have different interests or access authorizations, and *XML* enabled servers will need to know what data each user is allowed to have at a finer level of granularity than whole documents.

Current security technologies provide a number of security algorithms and technologies that can be used in *XML* security, but the actual formats used to implement security requirements are inappropriate for most *XML* security applications. For example, an *XML* document may be generated from various resources with varying security requirements due to its ability to express complex relationships between data. A user may like to limit the access to particular parts of an *XML* document with signatures. Because *XML* security is a key requirement, many efforts have been reported to deal with various security standards for *XML*, such as encryption and signature standards. *XML* digital signatures are designed for use in *XML* transactions. *XML* signatures have authentication, data integrity, and support for non-repudiation to the data that they sign. However, unlike non-*XML* digital signature standards, *XML* signatures have been designed to take advantage of the Internet and *XML*. *XML* Signatures are not limited to *XML* documents and can be applied to all types of electronic data, for example Hyper Text Markup Language (*HTML*) and Graphics Interchange Format (GIF) files. Compared to non *XML* digital signatures, it provides good privacy for the signers' signatures which can not be verified without interaction with the signer.

A complete solution to data security must meet three requirements: confidentiality, integrity and availability. Access control models and techniques provide high-assurance confidentiality and integrity. Access control plays an important role

in *XML* database systems. A large amount of effort has been reported that deals with various security standards for *XML*, such as encryption and signature standards. Access control models and mechanisms have also been widely investigated and several access control systems have been developed [42, 138, 18, 16, 38]. A standard access control model, known as Extensible Access Control Markup Language (XACML) [6], has also been developed which, however, has a limited set of features with respect to those of more advanced data models. On the other hand, traditional access control models are usually used for static authorization decisions based on the subjects' permissions on target objects. They have been used on the control of access to server-side objects. Traditional authorization decisions are generated at request time but do not consider ongoing controls for long access or for revocation. Recently proposed usage control [27] is a new access control model extending traditional access control models in multiple aspects. DTD and Schema level authorization in *XML* documents with usage control can be used to control *XML* documents in a dynamic environment. Objects-oriented database systems represent complex data structures and *XML* databases may be stored in objects-oriented database system. Therefore authorization models for *XML* databases are similar to the models for object-oriented databases. A new application begins with usage control.

Digital information sharing of multi-university environments usually occurs in broad, highly dynamic network-based environments, and formally accessing the resources in a secure manner poses a difficult and vital challenge. Dissemination of digital information has increased both commercial and non-commercial usage. The disseminative digital information should be managed by a distributor who can limit recipients' access to the information. There may be some undesirable leakages of digital information in E-Learning, and a secure protocol is required to control user

access to digital messages.

Role-based access control also has been widely used in database management. Many RBAC practical applications have been implemented since they can reduce administration cost and complexity [31, 35]. Permission-role assignments (*PRA*) is one important process in Role-based access control (*RBAC*) which has been proven to be a flexible and useful access model for information sharing in distributed collaborative environments. However, problems may arise during the procedures of *PRA*. Conflicting permissions may be assigned to one role, and as a result, the role with the permissions can derive unexpected access capabilities. The authorization algorithms are extended to the case of *PRA* with the mobility of a permission-role relationship.

This PhD project extends the current research that exists in *XML* databases and E-Learning. Five major tasks in this PhD research are focused on as follows.

1)Usage access control;

2)Usage access control in *XML* documents;

3)Access control in *XML* databases;

4)Access control in E-Learning;

5)Authorizations for permission-role assignments.

1.4 Organization of the dissertation

The dissertation consists of nine chapters. Their order is outlined and illustrated in Figure 1.2.

Chapter 2 gives the background which is necessary for a good understanding of

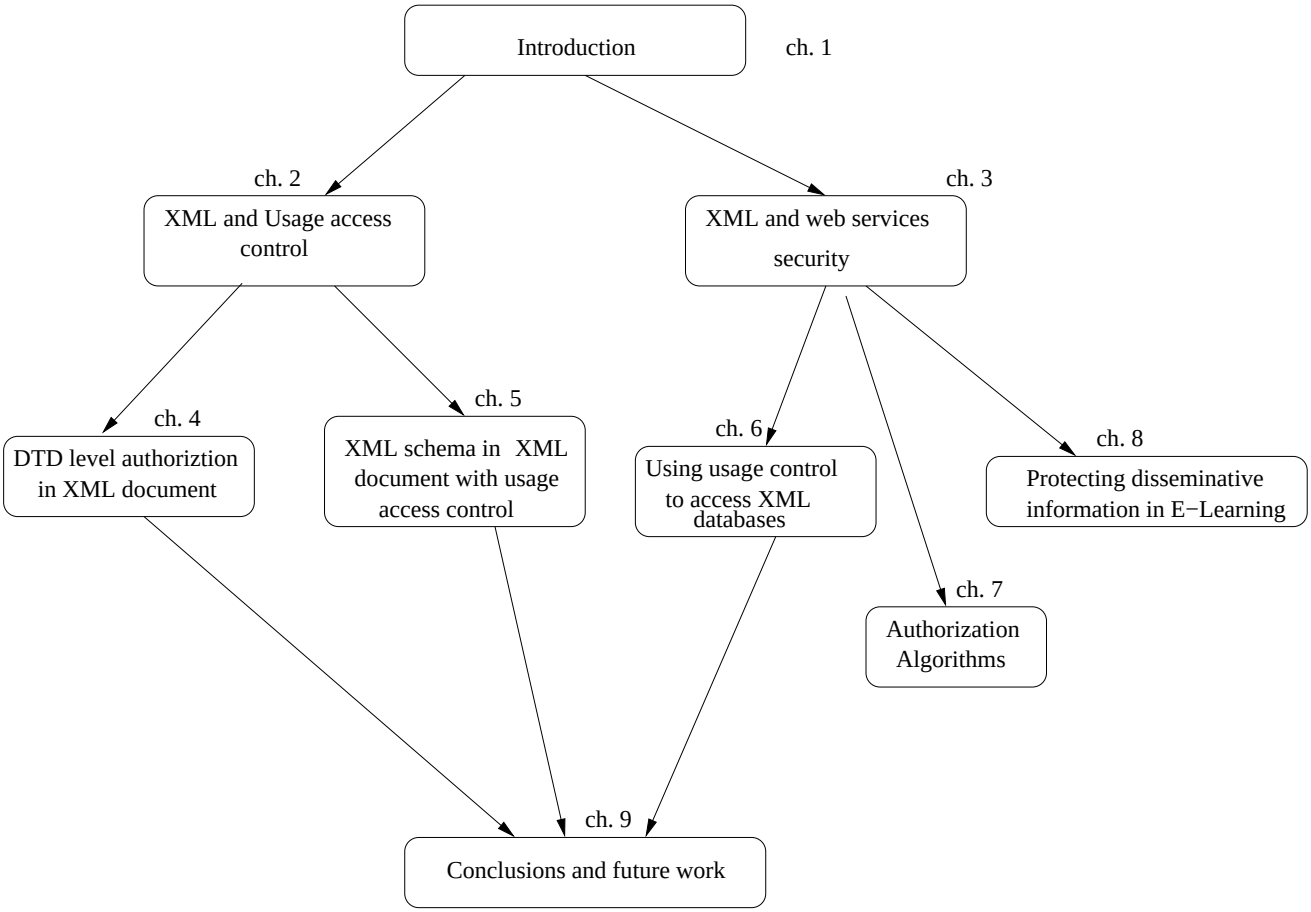

Figure 1.2: The structure of the thesis

this dissertation. First, it provides the basis of *XML* which is the data format considered in this study and two main validation specification mechanisms. Secondly it provides an overview of *XML* access control models. The focus is on usage access control models. Following that the study presents *XML* and web services security standards. This includes most specifications for these standards. From Chapter 4 to Chapter 8, the software technology plans are presented. In Chapter 4 and Chapter 5, DTD level and *XML* Schema authorization in *XML* documents with usage access control models are designed. Usage control models to access *XML* databases is developed in Chapter 6. In Chapter 7, the problems during the procedures of permission-role assignments in distributed collaborative environments are analyzed. In Chapter 8, a foundation for developing appropriate security solutions for dissemination of digital information is provided. Finally, conclusions and future work are drawn in Chapter 9.

Chapter 2

Background on underlying technologies

This chapter presents three different technologies. First, *XML* which serves as a versatile and universal format to share and distribute any kind of data. *XML* is a fundamental component in many web services and it is used to store and exchange data in the Internet environment. The basic *XML* technologies are introduced and described how *XML* and *XML* tree structure are defined. Second, a brief overview of access control models is provided. This includes four models: Discretionary Access Control (*DAC*), Mandatory Access Control (*MAC*), Role Based Access Control (*RBAC*) and Usage Access Control (*UAC*). Also access control system for *XML* documents are presented. Finally, the usage access control is addressed. Usage control has been considered as the next generation access control model with distinguishing properties of decision continuity. It has been proven to be efficient in improving security administration with flexible authorization management.

2.1 *XML* background

2.1.1 *XML*

Extensible Markup Language (*XML*) [24] is a simple, very flexible text format derived from Standard Generalized Markup Language (*SGML*) [33]. It is a standard for describing the structure of information and content on the Internet over the past several years. Originally designed to meet the challenges of large-scale electronic publishing, *XML* is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. *XML* is a markup language for documents containing structured information. Structured information contains both content (words, pictures, etc.) and some indication of what role that content plays. For example, content in a section heading has a different meaning from content in a footnote, which also means something different than content in a figure caption or content in a database table, etc. Almost all *XML* documents have the following features [24]:

- *XML* stands for EXtensible Markup Language,
- *XML* is a markup language much like *HTML*,
- *XML* was designed to describe data,
- *XML* tags are not predefined. You must define your own tags,
- *XML* uses a Document Type Definition (*DTD*) or an *XML* Schema to describe the data,
- *XML* with a *DTD* or *XML* Schema is designed to be self-descriptive.

The *XML* specification defines syntax and rules for using tags to structure information. A markup language is a mechanism to identify structures in a document. The *XML* specification defines a standard way to add markup to documents. Anyone can define a vocabulary of element tags and attributes to structure information of interest. Some rules defined in the *XML* specification may create "well-formed" *XML*. The well-formed *XML* may be processed with common *XML* tools. They may also explicitly define the structure of the documents they have defined, by creating an *XML* Schema or Document Type Definition (*DTD*). This allows *XML* documents to be validated. Therefore,

- *XML* is free and extensible.
- *XML* is a complement to *HTML*, but is not a replacement.
- *XML* is the future of Web services.

XML is a fundamental component in many *XML* web services. It overcomes the complexity of *SGML* and the user can define document structures, removing the limits of the fixed tags in Hypertext Markup Language (*HTML*). Comparing *XML* with *HTML* and *SGML*, *XML* was created so that richly structured documents could be used over the web [26]. *HTML* comes bound with a set of semantics and does not provide arbitrary structure. *SGML* provides arbitrary structure, but is too difficult to implement just for a web browser. Full *SGML* systems solve large, complex problems that justify their expense. Furthermore, *XML* languages created by different people may be combined. For example, if someone defines a language for expressing addresses, and we define one for purchase orders, we may use your address language within my purchase order language. To associate elements with the appropriate schemas and to avoid conflicting elements, *XML* namespaces are

used. An *XML* namespace is a collection of names. *XML* namespaces associate tags with unique identifiers (URIs) and are used to avoid ambiguity Namespaces [24]. A well-formed *XML* document is processed using general *XML*-aware tools, including parsers that understand the general rules of *XML* syntax and processing [128]. An advantage of *XML* is that the use of these tools does not depend on the specific vocabulary defined in a particular document. This means that once general tools have been created they may be used for many applications of *XML*. Another advantage is that it allows reuse of tools and training [24].

2.1.2 How is *XML* defined

XML is defined by a number of related specifications. For example, Extensible Markup Language (*XML*) 1.0 [24] define the syntax of *XML*. The *XML* specification is the primary focus of this article. *XML* Pointer Language (XPointer) [125] and *XML* Linking Language (XLink) [126] define a standard way to represent links between resources. In addition to simple links, like *HTML*'s < A > tag, *XML* has mechanisms for links between multiple resources and links between read-only resources. XPointer describes how to address a resource, XLink describes how to associate two or more resources. Extensible Style Language (XSL) [129] defines the standard stylesheet language for *XML*. As time goes on, additional requirements are being addressed by other specifications. Namespaces (dealing with tags from multiple tag sets), a query language (finding out what's in a document or a collection of documents), and a schema language (describing the relationships between tags, *DTD*s in *XML*) are all being actively pursued. Many *XML* languages have already been defined, including, *XHTML* [127] for creating web pages and for creating technical documentation, *RSS* [78] for content distribution (syndication), *RDF* [128]

for representing information, *MathML* [124] for mathematics markup, *BRML* [121] for business reports, and many others. The example in Table 2.1 displays library customer information in an *XML* document.

2.1.3 *XML* documents modeled as a tree structure

An *XML* document is a *tree* of nodes. Each node is the parent of zero or more child nodes. *XML* documents must contain a root element. This element is "the parent" of all other elements. The elements in an *XML* document form a document tree. The tree starts at the root and branches to the lowest level of the tree. All elements can have sub elements (child elements): The terms "parent", "child", and "sibling" are used to describe the relationships between elements. Parent elements have children. Children on the same level are called siblings (brothers or sisters). All elements can have text content and attributes (just like in *HTML*). Figure 2.1 shows the *XML* documents in tree structure. The root element in the example is <bookstore>. All <book> elements in the document are contained within <bookstore>. The <book> element has two children: <available> and <sold>.

XML documents not only show the content of data but also the constraints and relationships between data. In Table 2.1, the element *customerInfo* includes *bookstore* element and *books* sub-elements. The sub-element *description* is a simple type while sub-elements *available* and *sold* are combined with their own sub-elements. Since an *XML* document can express complex relationships between data, it may be generated from various resources with varying security requirements. In some situations a user may like to access the particular parts of an *XML* document. In the above *XML* document example, for the *textbook* objects everyone can read all information. However, some users' access to information such as *sold* and *buyer* will

```
<?xml version="1.0" encoding="UTF-8"? >
<customerInfo xmlns="http://www.bookstore.com/BooksInfo">
  <bookstore city="Toowoomba">
    <books >
      <available >
        <textbook>
          <description >
          Grade1 textbook
          </description >
        <price > $22.00 </price >
        <exercise book >
          <description >
          <English comprehensive>
          </description >
          <price > $18.00 </price >
        </exercise book >
      </available >
      <sold >
        < categorize > magazine </categorize>
        <price > $30.00 </price >
        <buyer >
          <name > Tony </name >
          <address > Jilan street, 5 </address >
          <city > Toowoomba </city >
        </buyer >
      </sold>
    </books >
  </bookstore >
</customerInfo >
```

Table 2.1: XML document example

be restricted. This is because when an internal or external user accesses this document, his/her access permission has to be limited according to security policies in all

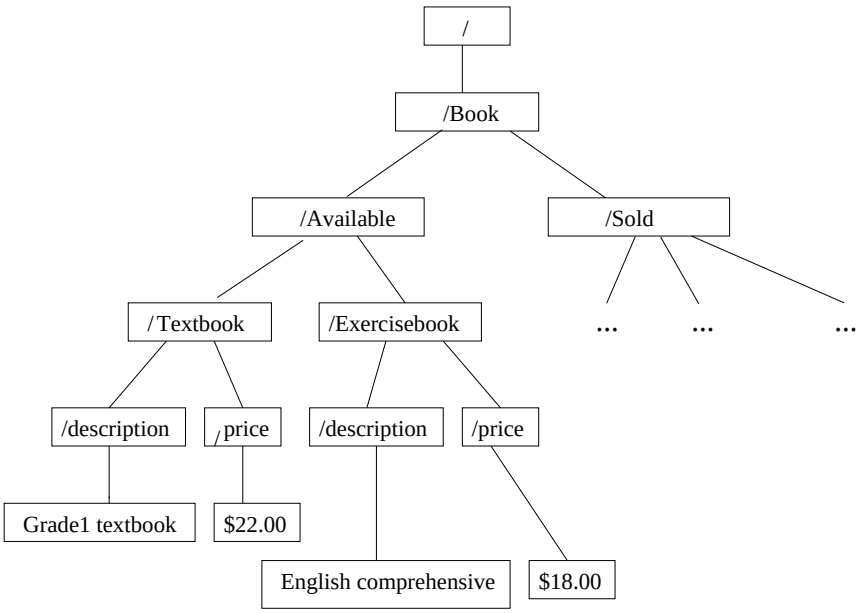

Figure 2.1: XML tree structure

databases. This example shows that securing *XML* documents forms a significant topic for research.

XML is a markup language for describing semi-structured information. An *XML* document consists of elements, attributes and text nodes, each delimited by a pair of start and end tags (e.g. <price> and </price>) or by an empty tag. The content of each element is a sequence of elements or text nodes. An element has a set of attributes, each of which has a name and a value.

2.1.4 *DTD* and *XML* documents

XML documents can be classified into two categories: well-formed and valid [12]. A document is said to be well-formed if it follows the grammar rules of *XML*, such that there is exactly one element that completely contains all other elements, and elements may nest but not overlapped, etc. A well-formed document is valid only if it contains a proper *DTD* in the source and if the document obeys the constraints of that declaration. Validation requires an *XML* instance to contain specified elements

and attributes. On the other hand, since *XML* is a structural transformation, it can transform one structure to another structure. As a Document Markup Language, *XML* can be distinguished from *HTML* (the most widely used markup language for web applications) in the customized set of tags it provides (which can convey the semantics of the data represented) as compared to the fixed tag set of *HTML*. Hence an *XML* document can provide a logical representation of its data contents. The logical organization of the tags themselves, that are used in an *XML* document, are provided in a separate document called the Document Type Definition or the *DTD*.

In general, we identify two levels, instance and *DTD* (the Document Type Definition) at which authorizations on *XML* documents can be defined. *XML* documents and *DTD*s naturally support two levels of authorization. 1). Low-level authorizations, associated with *XML* documents, providing full control of authorizations on a document by document basis; 2). High-level authorizations, associated with *XML DTD*s, providing structure and element declarations of access permissions. Different requirements may have a call for the support of access restrictions at the level of each specific document [74]. In the access control model the central authority uses *XML DTD*s to specify the format of information to be changed.

Document Type Definition (*DTD*) and *XML* Schema are two main validation specification mechanisms [117, 118]. A *DTD* contains a formal definition of a particular type of *XML* document. A *DTD* consists of two parts: the element declarations and the attributes declarations. Elements are the most important components of an *XML* document. Element declarations in the *DTD* specify the names of elements and their contents. They also describe sub-elements and their cardinality. Attributes represent properties of elements. Attribute declarations specify the attributes of each element, including their name, type, etc. Attributes can be marked

as *required, implied*, or *fixed*. Attributes with *required* must have an explicit value for each occurrence of the associated elements. Attributes with *implied* are optional. Attributes with *fixed* have a fixed value. Entities are used to include texts and binary data. Notations specify how to manage entities and binary data. Entities and notations are not considered in this paper since both of them are used to describe the physical structure of an *XML* document.

However, the *XML* documents corresponding to a *DTD* must obey a structure defined by that *DTD*. Each *DTD* is a schema and *XML* documents corresponding to that *DTD* are instances of that schema. But the *DTD* structure is not restricted. For instance, two different *XML* documents with the same schema may widely differ in the elements. As a well-formed *XML* document is in a nested structure, there are some languages (e.g., X*HTML* and *XML* DOM) which are applied to locate elements with patterns. An *XML* document can be generated from various resources to fit applications with different structures. These main aspects about *XML* are discussed in [43]. The example in Table 2.2 displays an *XML DTD* for a corresponding valid *XML* document in Table 2.1.

2.1.5 *XML* Validation: DVD and schema

As the two main validation specification mechanisms for *XML* documents, Document Type Definition (*DTD*) and *XML* Schema [24, 42] have their own features. *DTD* is the first and earliest language to define the structures and content of *XML* documents. A *DTD* is a file which contains a formal definition of a particular type of *XML* documents. It contains or points to markup declarations that provide a grammar for a class of documents. Each *DTD* is also a schema and *XML* documents corresponding to that *DTD* are instances of that schema. But it has some

```
<?xml version="1.0" encoding="UTF-8"? >
  <xs:annotation >
    <xs:documentation >
      Customer Information Instance
    </xs:documentation >
  </xs:annotation >
  <xs:ELEMENT bookstore (book+) >
  <xs:ELEMENT books (available*, sold*) >
  <xs:ELEMENT available (catalog, price, exercise book*, buyer) >
  <xs:ELEMENT sold (categorize, price, buyer) >
  <xs:ELEMENT exercise book (PCDATA) >
  <xs:ELEMENT buyer (name, address, city, discount?) >
  <xs:ELEMENT categorize (PCDATA) >
  <xs:ELEMENT price (PCDATA) >
  <xs:ELEMENT description (PCDATA) >
  <xs:ELEMENT name (PCDATA) >
  <xs:ELEMENT address (PCDATA) >
  <xs:ELEMENT city (PCDATA) >
  <xs:ELEMENT discount (PCDATA) >
  <xs:ATTLIST bookstore city CDATA # REQUIRED >
```

Table 2.2: XML DTD example

limitations: firstly, a *DTD* file is neither a well-formed nor a valid *XML* document. It is difficult to specify constraints on structures and content of *XML* instances with *DTD*s. Secondly, it is hard to handle the name confliction in *DTD*s. For example if a student is involved in two departments, the two course names in each department will conflict if a *DTD* is applied by both of them. Finally, a *DTD* can not define datatypes, which makes it difficult to use by other *DTD*s. On the other hand, an *XML* Schema is an *XML*-based alternative to *DTD*. An *XML* Schema describes the structure of an *XML* document. An *XML* Schemas express shared vocabularies and allow machines to carry out rules made by people. They provide a means for defin-

ing the structures, content and semantics of *XML* documents [91]. *XML* Schemas are extensible to future additions. *XML* Schemas are richer and more powerful than *DTD*s. *XML* Schemas are written in *XML*. *XML* Schema is an *XML* document itself. It supports data type and namespaces. Complex user-defined datatypes can be created in *XML* Schemas. Namespace is supported in *XML* Schemas to solve name conflictions. For these reasons, *XML* Schemas are a richer and more powerful way of describing information than what is possible with *DTD*s. Very soon, *XML* Schemas will be used in most Web applications as a replacement for *DTD*s. Since *DTD* is not *XML* well-formed and valid-formed, the access control policy on *XML* instance documents and *DTD* have to be implemented separately. By using schemas, we can define and enforce the permissions on schema objects and instance objects with a uniform mechanism. The example in Table 2.3 displays an *XML* Schema for a corresponding valid *XML* instance in Table 2.1.

An *XML* Schema defines elements and attributes that can appear in a document. It also defines which elements are child elements, the order of child elements and the number of child elements. It provides data types for elements and attributes, default and fixed values for elements and attributes.

2.2 Background on access control

Access control is the ability to permit or deny the use of a particular resource by a particular entity [59]. Access control mechanisms can be used in managing physical resources (such as a movie theatre, to which only ticket holders should be admitted), logical resources (a bank account, with a limited number of people authorized to make a withdrawal), or digital resources (for example, a private text document on a computer, which only certain users should be able to read)[74].

```
<?xml version="1.0" encoding="UTF-8"? >
 <xs:schema>
  targetNamespace="http://www.bookstore.com/BooksInfo
  xmlns:xs= "http://www.w3.org/2001/XML Schema
  elementFormDefault="qualified>
   <xs:annotation>
    <xs:documentation >
     <Bookstore Information Instance>
    </xs:documentation >
   </xs:annotation >
  <xs:element name="bookInfo">
   <xs:sequence>
    <xs:element name="books" type="xs:string"/>
     <xs:element name="available"/ >
      <xs:element name= "catagorize" type="xs:string"/>
      <xs:complexType>
       <xs:sequence>
        <xs:element name="exercisebook"/>
         <xs:sequence>
          <xs:element name="description" type="xs:string"/>
          <xs:element name="price" type="xs:string"/>
         </xs:sequence>
       </xs:sequence>
      </xs:complexType>
   </xs:sequence>
  <xs:sold>
   <xs:sequence>
    <xs:element name="categorize" type="xs:string"/>
    <xs:element name="price" type="xs:string"/>
    <xs:element name="buyer" type="buyerInfoType"/>
   </xs:sequence>
```

```
    <complexType name="buyerInfoType">
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
      </xs:sequence>
    <complexType>
</xs:schema>
```

Table 2.3: XML schema example

Access control plays a crucial role in Web database management. In particular, access control must be tailored to the language used to describe the data and to the types of actions that can be executed on such data, such as browsing. Therefore, developing an access control model and related mechanisms in terms of *XML* is an important step. An access control is a means of obtaining data confidentiality specifying which users can perform which actions on which data [74]. Many access control models have been defined so far to simplify and secure the management of the access control rights. The four most best known models are as follows: Discretionary Access Control (*DAC*), Mandatory Access Control (*MAC*), more recently Role Based Access Control (*RBAC*) and Usage Access Control (*UAC*).

Discretionary Access Control (*DAC*) is a means of restricting access to information based on the identity of users and/or membership in certain groups [77]. For *DAC*, access decisions are typically based on the authorizations granted to a user based on the credentials he/she presented at the time of authentication (user name, password, hardware/software token, etc.). In most typical *DAC* models, the owner of information or resource is able to change its permissions at his/her discretion. *DAC* models allow subjects to grant authorizations on the data or other subjects. Because of such flexibility, discretionary policies are adopted in many application en-

vironments. Usually the commercial database management system (*DBMS*) adopts such policies. However, the organization generally regulates the way in which a user is authorized to pass the access rights onto the other subjects.

Mandatory Access Control (*MAC*) refers to a type of access control by which the operating system constrains the ability of a subject or initiator to access or generally perform some sort of operations on an object or target. It is usually appropriate for extremely secure systems including multilevel secure military applications or mission critical data applications [101]. *MAC* mechanisms assign a security level to all information, assign a security clearance to each user, and ensure that all users only have access to the data for which they have a clearance. The subject clearance level specifies the level of trust. An object's sensitivity specifies the level of trust required for access. In order to access a given object, the subject must have a clearance level equal to or higher than the requested object sensitivity. With mandatory access control, this security policy is centrally controlled by a security policy administrator. Users do not have the ability to override the policy, for example, by granting access to files that would otherwise be restricted. By contrast, discretionary access control (*DAC*), which also governs the ability of subjects to access objects, allows users the ability to make policy decisions and/or assign security attributes. The traditional Unix system of users, groups, and *rwx* permissions is an example of *DAC*. *MAC*-enabled systems allow policy administrators to implement organization-wide security policies. Unlike with *DAC*, users cannot override or modify this policy, either accidentally or intentionally. This allows security administrators to define a central policy that is guaranteed (in principle) to be enforced for all users.

Role-Based Access Control (*RBAC*) is an approach to restricting system access to authorized users. It is a newer alternative approach to mandatory access control

(*MAC*) and discretionary access control (*DAC*) [107, 31]. *RBAC* is a flexible access control technology sufficiently powerful to simulate Discretionary Access Control (*DAC*) and Mandatory Access Control (*MAC*).

In *RBAC*, access decisions are based on an individual's roles and responsibilities within the organization or user base. The process of defining roles is usually based on analyzing the fundamental goals and structure of an organization and is usually linked to the security policy. For instance, in a medical organization, the different roles of users may include the roles of doctors, nurses, attendants, patients, etc. Obviously, these members require different levels of access in order to perform their functions, but also the types of web transactions and their allowed context vary greatly depending on the security policy and any relevant regulations.

Usage control is a new access control model which extends traditional access control models and other access control models in many aspects. The term usage means usage of rights on digital objects. The main different properties of usage control with traditional access control models are continuity of access decision and mutability of subject attributes and object attributes.

Although *DAC* and *MAC* offer a high level of security, both of them have drawbacks. For *DAC*, the administrators are not able to centrally manage permissions on files/information stored on the web server. *MAC* is too rigid and cannot handle exceptions; for example, a user with a certain level of security can not as an exception have access to data requiring a higher level security. *RBAC* provides a way to define access rights just by assigning roles, object hierarchies and constraints. For the sake of simplicity we have chosen to rely on the *DAC* model in our study.

2.2.1 Access control system for *XML*

In recent years more and more information is made in *XML* format, especially data on Intranets and on the global Net. Developers and end-users have raised their concerns about *XML* security problems. However, early research work about *XML* has not related to access control and security because *XML* was initially used as a data format for documents. But the way securing documents apply to *XML* data has caused some problems if additional measures are necessary. For example users are able to control their interaction on web sites by getting the information they are interested in from the dynamically generated *XML* documents. However, different users may have different interests or access authorization. Some users will need to allow access or block the entire *XML* instances, while others will control access portions of documents. Therefore, a model for dynamic access control with granularity control is needed.

With authorization, the server will know what information can be sent to the user based on that user's identity or certified documents. Encryption will only provide the message to the users with adequate decryption keys. Therefore, *XML* security should support fine grain granularity. The following shows the basic requirements for standardizing *XML* access control at the tag level [69].

1. Support of authorizations at different organizational levels. Security policies are able to be used either on huge document based data sources or on single documents.

2. Fine-grained access control. Access control policies should be supported at all levels of granularity, including documents and individual *XML* elements.

3. Extension to existing Web server technology. *XML* documents are usually used for Web sites. Using *XML* access control would not affect existing APIs and development tools.

4. Integration with existing authorization technologies such as digital signatures. *XML* access control should complement authorization based on digital signatures.

5. Transparecy. The access control system should be transparent to the requesters. In particular, access control should be validity with their *DTD*s.

Figure 2.2 depicts the *XML* access control system in the network. A central authority uses *XML DTD*s to specify the format of information to be exchanged within the network. *XML* documents instances of such *DTD*s are defined and maintained at each site. The schema-instance relationship between *XML* documents and *DTD*s support the two levels of authorization(low-level authorizations, high-level authorizations). Both of them allow for fine grained specification. Each local authority managing a Web site retains the right to define its own authorizations for its documents. It can also define authorizations at *DTD* level.

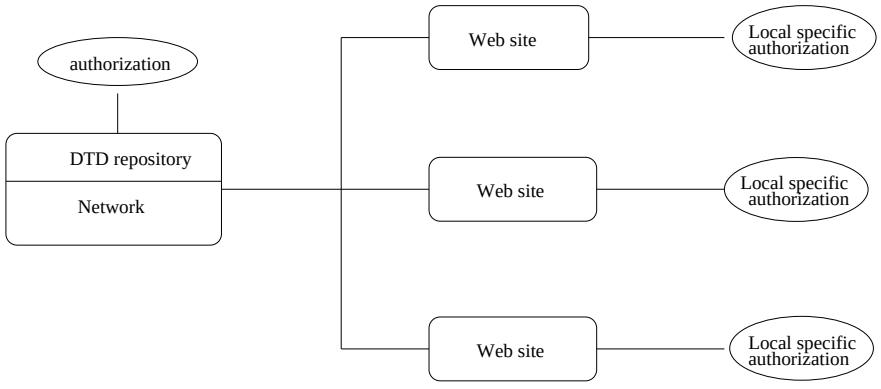

Figure 2.2: XML access control system

Because requiring the specification of authorizations for each single document would make the authorization specification task much too heavy. Using *DTD* could solve this problem by allowing protection requirements to refer to *DTD* or *XML* documents, where requirements specified at the level of *DTD* apply to all those document instances of the *DTD*s. In the network environment, there are two types of *DTD*-level authorizations, *Global DTD-level authorizations* and *Local DTD-level authorizations*. Global *DTD*-level authorizations managed by a central authority can be effectively used to implement corporate-wide access control policies on document classes. Local *DTD*-level authorizations specified by departmental authorities allow for department-wide access control policies complementing the corporate ones [41].

Furthermore, the elements and attributes within an *XML* documents can be used to specify authorizations at fine grained level. In other words, authorizations specified for an element apply to all its attributes. To avoid the need of specifying authorizations for each signal element in a document, the document structure can support a recursive interpretation of authorization by which an authorization specified on an element applies to its whole content (subelements and attributes). In our models, authorizations allow for the specification of an element or apply recursively to all its subelements.

Authorizations specified for each *XML* document/*DTD* (elements within) are stored in (*XML* Access Sheets - XAS) associated with the document/*DTD*. Figure 2.3 shows where *XML* authorization information is stored in XAS.

In access control processing, *DTD*-level authorizations specified at the global level or at the local level depend on each *DTD*. Organization-wide authorizations apply to all the documents in the network while site-specific authorizations apply only to documents stored at the site. However, organization-wide and site specific

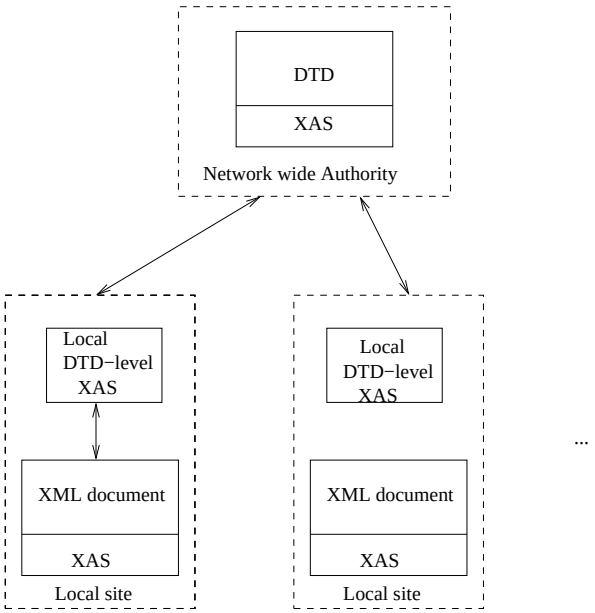

Figure 2.3: Authorization information store

authorizations are treated in the same way.

Figure 2.4 shows the structure of an access control model for *XML* documents. A user requests access to resources in a system, and the system then determines whether to admit or reject by referring to information on access control policy and *XML* documents requested after confirming that the requester is a legitimate user.

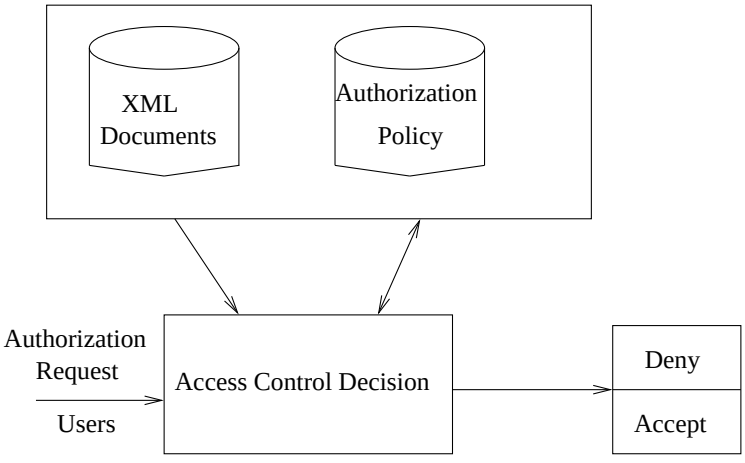

Figure 2.4: XML document access control architecture

2.3 Usage access control

The classic access matrix model has stood fundamentally unchanged for over three decades. The core concept of the access matrix is that a right (or permission) is explicitly granted to a subject to access an object in a specific mode, such as read or write. The core idea has been successfully elaborated in different directions in the familiar models of discretionary, mandatory and role-based access control, to accommodate a diverse range of real-world access control policies [95, 141].

In recent years, many researchers have realized that the access matrix needs fundamental enhancements to meet the needs of modern applications and systems. These researchers have provided new concepts such as trust management, digital rights management (*DRM*), task-based access control, provisional authorization, obligations and more. However, such access control normally deals only with authorization decisions on users' access to target resources. A new approach to access control called usage control is a fundamental enhancement of the access matrix. Usage control is a generalization of access control. It enriches and refines the access control discipline in its definition. As the core model of usage control, a family of *ABC* models is built around three decision factors, authorizations (A), obligations (B) and conditions (C). The familiar notion of authorization is based on subject and object attributes. Obligations require some action by subject. Conditions are environmental or system-oriented factors that predicate access. Further, the *ABC* model has mutable attributes. It also recognizes the continuity of access enforcement so the decision to allow access is not only made prior to access, but also during the time interval. *ABC* is the first model to address a systematic and comprehensive extension of the classic access matrix. By integrating authorization, obligations and conditions along with mutable attributes and ongoing enforcement, it encompasses

traditional discretionary, mandatory and role-based access control. It also encompasses emerging applications such as trust management, digital rights management and so on. *ABC* is a core model of usage control in that it focuses on the process of access enforcement.

In traditional access control, an authorization decision is made based on subject attributes, object attributes, and requested rights. Attributes include identities, capabilities, or properties of subjects or objects. Figure 2.5 shows this attribute -based traditional access control.

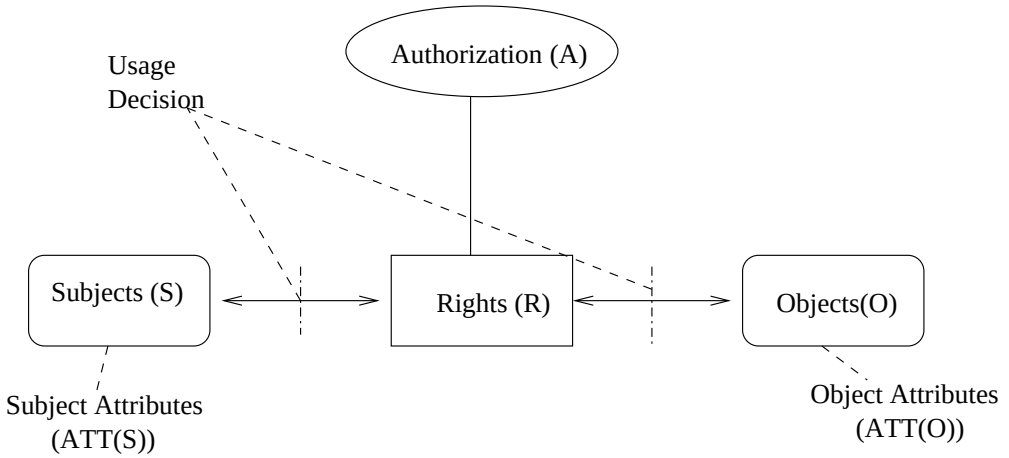

Figure 2.5: Tradtional access control

Although this attributed-based approach of traditional access control can cover many applications, the digital information systems require more than classical authorizations. For instance, sometimes a usage decision is based on fulfilment of required actions, rather than existence of subject attributes and object attributes. This decision factor is called an "obligation". In addition to authorization and obligation, there are certain situations where access or usage needs to be limited due to a certain environmental or system status. For this requirement, this usage decision factor is called a "condition" and is required together with authorization and

obligation for access control.

The *ABC* model is a core model for usage control. There are eight core components in this usage control model: subjects, subject attributes, objects, object attributes, rights, authorizations, obligations, and conditions [95, 141] (see Figure 2.6). In usage control, subjects and objects are familiar concepts with traditional access control. A right represents access of a subject to an object, such as read or write. The existence of the right is determined at the point in time when the access is attempted by the subject. Obligations and conditions are new concepts that can resolve certain shortcomings in traditional access controls. Obligations are requirements that have to be fulfilled by obligation subjects for allowing access. Conditions are subject and object independent environmental or system requirements that have to be satisfied for access.

The usage decision functions indicated in Figure 2.6 make this determination based on subject attributes, object attributes, authorizations, obligations and conditions at the time of usage requests.

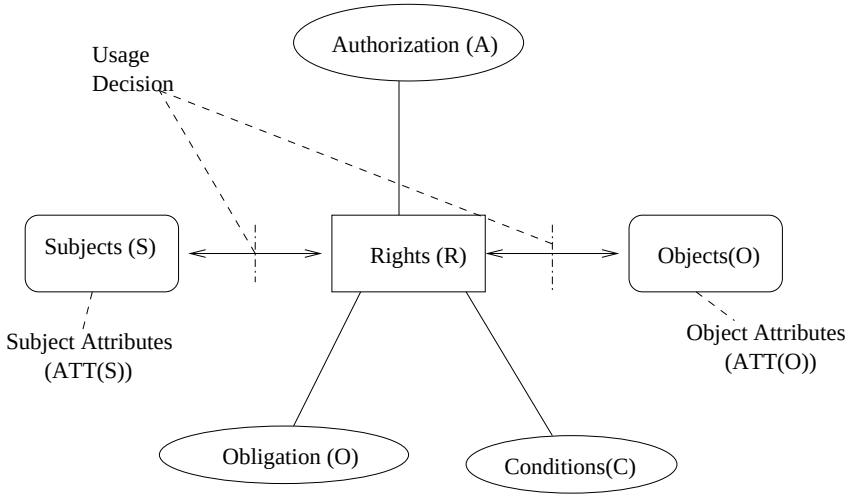

Figure 2.6: Components of usage control model

Subjects, objects, and rights can be divided into detailed components with dif-

ferent perspectives. Subjects are entities associated with attributes, and they hold and exercise certain rights on objects. A subject can be a user, a group, a role, or a process. In the usage control model, the subjects can be *consumer subjects (CS)*, *provider subjects (PS)*, or *identifier subjects (IS)*. Consumer subjects are entities that receive rights and objects and use these rights to access the objects. Provider subjects are entities that provide an object and hold certain rights to it. The identifier subjects are entities that are identified in digital objects that include their privacy-sensitive information. Objects are entities that subjects hold rights to, whereby the subjects can access or use objects. Objects are also associated with attributes, either by themselves or together with rights.

Rights are privileges that subjects can hold on objects. Rights consist of a set of usage functions that enables a subject's access to objects. The authorization of rights requires associations with subjects and objects. Like subjects, rights can also be divided into *consumer rights (CR)*, *provider rights (PR)*, and *identifier rights (IS)*. The rights include rights for access and use of objects and rights for delegation of rights. In this research, the focus is on rights for access. In usage access control rights can be divided into many functional categories. The two main fundamental rights categories might be a view and modification. Modification includes a change to an existing digital object and the creation of a new object that reuses an original digital object.

Subject and object attributes can be used during the access decision process. Subject attributes are identities, group names, roles, memberships, security clearances and so on. For example, an online shopper and a university student in a management system can be subjects. Object attributes are associated with objects such as security labels, ownerships, classes, access control lists and so on. For in-

stance, in an on-line shopping store a price could be an object attribute, where an electron blood pressure measure is priced at $120 and with delivery costs $135.

Authorizations, obligations and conditions are decision factors used to check and determine whether a subject should be allowed to access an object. Authorizations are based on subject and object attributes and specific rights. Authorizations can be either pre-authorization (preA) or ongoing-authorization(onA). Pre-authorization is performed before authorization is required to the access. In general, the authorization of most traditional access control is assumed to be done before access is allowed (pre). But ongoing authorization may be performed during the access, such as a bookstocking list in a bookstore which may be periodically checked while the access is in progress. An access is immediately revoked if the relevant item's number becomes 0 when it appears on the list. However, it is quite reasonable to extend this for continuous enforcement by evaluating usage requirements throughout usages (ongoing). Authorizations may require updates on subject and object attributes. These updates can be either 'pre', 'ongoing', or 'post', and they are called continuity properties. Figure 2.7 shows the continuity properties in a usage control model.

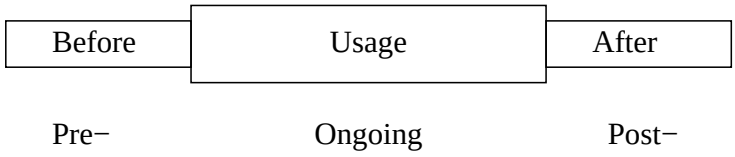

Figure 2.7: Continuity properties of usage control

Obligations are requirements that a subject must perform before (pre) or during (ongoing) access. In real world implementation, this may have to be done by agreeing on the fulfilment of obligations before obtaining the rights and at the time when obligation-related authorization rules are checked. An example of a pre-obligation

is the requirement that a user must provide some contact and personal information before accessing the IEEE digital library. The requirement that a user has to keep certain advertising windows open while accessing a particular service, is an example of an ongoing obligation. Subject and object attributes can be used to decide what kind of obligations are required for access approval. Traditional access control has hardly recognized the obligation functions though many of them implement the obligation functions partially and implicitly.

Conditions are decision factors that depend on environmental and system-oriented requirements. The system should verify during the authorization process based on authorization rules before allowing usage of rights to a digital object. There are two types of conditions, one is *Dynamic conditions*, and the other is *Static conditions*. Dynamic conditions have information that may have to be checked for updates at each time of usage. Static conditions have information that does not have to be checked for updates. Subject and object attributes can be used to select which condition requirements have to be used for a request. For example, IEEE member can access full papers in the IEEE digital library. Subject and object attributes can also include the security status of the system, such as low level, normal, high alert.

2.4 Conclusions

This chapter we gave an overview of *XML* standard and the two main validation specification mechanisms Document Type Definition (*DTD*) and *XML* Schema. Access control and an access control system for *XML* were introduced. Meanwhile, a next generation access control model, usage access control with distinguishing properties of decision has been presented. Some technology topics in *XML* documents and *XML* databases with usage access control models will be analyzed in detail in

the rest of the dissertation.

Chapter 3

XML and web services security

This chapter presents an introduction to *XML* documents, then analyses *XML* security technologies including *XML* digital signature, *XML* encryption and key management. *XML* security technologies are essential and adopted for Web services. Finally *XML* security languages and web service security are discussed.

The information in this chapter is based on a published paper [120].

3.1 Introduction

With an increasing amount of semi-structured data, *XML* documents have been widely used in a web environment. *XML* has become important. Several specifications progressed toward providing a comprehensive standards framework for securing *XML*-based applications have been presented. These applications can be effective to protect information on a website. In this chapter, we present *XML* and Web service security main standards and most specifications for these standards. Each standard which connects with protecting *XML* based documents is discussed. In particular, we present *XML* undeniable signature as an application with an *XML* digital signature. Relations between these standards based on existing technologies

are also briefly described. Finally, comparisons with related works are analysed.

This chapter is organized as follows: in section 2, the basic *XML* documents are introduced. Sections 3, 4 and 5 present a brief overview of the following core *XML* Security standards [92]:

- Integrity and signatures - *XML* Digital Signature
- Confidentiality - *XML* Encryption
- Key Management - *XML* Key Management Specification
- Authentication and Authorization Assertions - Security Assertion Markup Language
- Authorization Rules - *XML* Access Control Markup Language
- Web Services Security - WS-Security

Finally the conclusions are presented in section 6.

3.2 *XML* documents

XML Web services are a successful paradigm for the rapid development of many complex web-based applications [37]. *XML* is an extensible markup language that differs from HTML since it allows users to define their own vocabularies of opening and closing tags. The following example shows an *XML* document of student card information which specifies datatypes and relationships. In this example a student's information in an *XML* document can be accessed by anyone if there is no protective security.

```
<?xml version="1.0" encoding="UTF-8"? >
<studentInfo XMLns="http://www.school.com/StudentInfo">
 <ID> 123-45-6789 </ID>
    <name>
       <firstName > Mike </firstName>
       <lastName > Harman </lastName>
    </name>
    <studentCardInfo>
       <type> Student card </type>
       <cardNo> 8888888888888 </cardNo>
       <expireDate> 10/08 </expireDtae>
       <nameOnCard> Mike Harman </nameOnCard>
    </studentCardInfo>
</studentInfo>
```

Table 3.1: An XML document for a student card

In Table 3.1, the element *studentInfo* includes *ID*, *name* and *studentCardInfo* sub-elements. The sub-element *ID* is a simple type while sub-elements *name* and *studentCardInfo* are combined with their own sub-elements. In this example, for the *studentInfo* objects everyone can read all the information. On the other hand, when an internal or external user accesses this document, his/her access permission has to be limited according to security policies in the databases.

3.3 *XML* digital signature

Digital signatures are an important element in electronic security because they can be used to ensure the integrity, authenticity, and non-repudiability of data [112]. Commonly used digital signature schemes, such as RSA and DSA, employ a pair of mathematically related keys. Each person wishing to create digital signatures must have a unique key pair (private key and public key). Normally the private key is

kept in secret and used for signing a document while the public key is made public and used for verifying its owner's signatures.

XML digital signatures are designed for use in *XML* document transactions. *XML* signatures include authentication, data integrity, and support for non-repudiation to the data that they sign. It has established an approach for data exchange as well as some vocabulary shared by the other standards. The *XML* digital signature specification on the website, www.w3.org/TR/2002/REC-xmldsig-core-20020212 [3], is a final draft which provides guidelines in this area. It specifies the basic structures of defining a digital signature using *XML*. This specification identifies methods for referencing collections of resources, algorithms, and keying and information management. *XML* Signatures can be applied to any digital contents, including *XML* documents. Table 3.2 shows the structure of *XML* signature and its key elements.

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

Table 3.2: XML signature structure

An *XML* digital Signature can sign more than one type of resources [53]. There are three types of *XML* digital signature structures, namely enveloped signature,

enveloping signature and detached signature. For enveloped and enveloping signatures, the signed *XML* documents and their securing signatures are within the same files. For the detached signatures, the *XML* signatures are in a separate document. The three types of *XML* Signatures are discussed below.

1. Enveloped signature structure:

   The *XML* signature is included in the *XML* document. It is a child element of the *XML* document. The data being signed envelops the <signature> and </signature> tag pair.

2. Enveloping signature structure:

   The *XML* document is included in the *XML* signature. It is a child element of the *XML* signature. The data being signed is inside the <signature>, </signature> pair as an object element.

3. Detached Signature structure:

   The *XML* Signature is in a separate document from the signed *XML* document. The location of the signed document is referenced in the *XML* Signature. This type of signature is used for non-*XML* documents

Table 3.3 is an example of enveloped signature in an *XML* document where a signature is added to the <customerInfo> element, and the <Signature> element would be a child of the <customerInfo> element.

An *XML* digital signature differs from other protocols for message signing, such as PGP, since it supports signing only specific portions of the *XML* tree rather than the complete documents.

```
<customerInfo xmlns="http://www.hotel.com/CustomerInfo">
  < ID > 123-45-6789 <ID >
  <name >
    <firstName > Tony </firstName>
    <lastName > Zhang </lastName>
  </name >
  <creditCardInfo>
    <type > Master card </type >
    <cardNo > 8888888888888 </cardNo >
    <expireDate > 10/07 </expireDtae >
    <nameOnCard > Tony Zhang </nameOnCard >
  </creditCardInfo >
  <Signature
    xmlns="http://www.w3.org/2000/09/xmldsig#"
    ....
  </Signature >
</customerInfo >
```

Table 3.3: XML enveloped signature

3.3.1 *XML* undeniable signature

The central role of digital signatures in the commercial and legal aspects of the evolving electronic commerce world are well recognized [114]. Digital signatures bind signers to the contents of the documents they sign. The ability for any third party to verify the validity of a signature is usually seen as the basis for the "non-repudiation" aspect of digital signatures, and is their main source of attractiveness. Advanced signature schemes include group signatures, blind signatures, undeniable signatures and proxy signatures. Undeniable signatures were firstly introduced by Chaum and Van Antulerpen [30]. They secure that signatures cannot be easily verified. Undeniable signature schemes are used in places where the co-operation of

a signer is required in the verification [30]. An *XML* undeniable signature approach builds a bridge between the existing *XML* technologies and data security theories. The *XML* undeniable signature scheme is based on the RSA undeniable signature algorithm for *XML* documents.

An undeniable signature algorithm for <SignatureMethod> is applied in the *XML* signature structure. This consists of generating a signature, confirmation protocol and deniable protocol. The application of a undeniable signature algorithm in the *XML* signature is a fundamental and important step for the new *XML* undeniable signature. The *XML* undeniable signature approach is a new way to secure sensitive information in *XML* document transitions and signers can not deny their signatures. This can provide a secure framework for *XML* web services.

3.4 *XML* encryption and key management

3.4.1 *XML* encryption

The *XML* Encryption Syntax and Processing specification defines an *XML* vocabulary and processing rules for protecting confidentiality of *XML* document [4]. It may work in whole or parts of *XML* documents and non-*XML* data as well. In contrast to other commonly used technologies for confidentiality such as SSL, *XML* encryption also applies to document parts and documents in persistent storages.

In order to make content confidential, the owner of the content may use an encryption mechanism to do it. This process will make the content unintelligible until it is decrypted. Encryption generally uses symmetric key encryption, since this is an efficient technique even for large documents. Symmetric key encryption uses the same key for both encryption and decryption. But this may cause a problem as send-

ing confidential information to a receiver, the sender and the recipient must share the symmetric key without involving anyone else. This can be difficult without person to person contact. To avoid this problem and make it easier to share confidential content with a number of people, asymmetric or public-key cryptography was designed. Public key cryptography uses a matched pair of keys, one for encryption and one for decryption. In this encryption process the sender encrypts using the recipient's public key that can be shared widely. Recipient decrypts using a private key that is known only to themselves. This helps to take over the difficulty of establishing confidential communication. But when public key cryptography and symmetric cryptography are used together, they will become more efficient. The symmetric key is used to encrypt the content, and then the symmetric key is encrypted using public key cryptography. Both the encrypted content and encrypted symmetric key are then sent to the recipient. In the Encryption syntax, the <EncryptedData> element is the core element. It also contains: <EncryptionMethord>, <KeyInfo>, <CipherData> and <EbcryptionProperties> sub-elements. Similar to the above example, Table 3.4 shows a *customerInf* in an *XML* document. It only encrypts the elements of the <CreditCard> element.

3.4.2 *XML* key management specification

Public key technology is an essential part of *XML* Digital Signature, *XML* Encryption and other security applications [8]. *XML* Key Management Specification (*XKMS*) defines protocols between *XKMS* client and server for performing public-key infrastructure (*PKI*) operations [2]. *XKMS* defines *XML* message formats to support requests and responses for public key management, including public key registration, public key validation, public key discovery and public key revocation.

```
<customerInfo xmlns="http://www.hotel.com/CustomerInfo">
< ID > 123-45-6789 </ID >
<name >
   <firstName > Tony </firstName>
   <lastName > Zhang </lastName>
</name >
<creditCardInfo>
   <EncryptedData
   XMLns="http://www.w3.org/2001/04/xmlenc#"
   Type="http://www.w3.org/2001/04xmlenc#Content">
      <CipherData>
         <CipherValue>A12B34C657</CipherValue>
      </CipherData>
   </EncryptedData>
</creditCardInfo>
</customerInfo>
```

Table 3.4: XML encryption

Thus, *XKMS* is designed to be used along with *XML* digital signature and *XML* encryption, and it helps to manage the public key enabling signature verification and encrypting for recipients. *PKI* plays an important role in Web services and E-commerce. Since *PKI* operations are too expensive for small devices, using *XKMS* may reduce the processing burden by moving it to an *XKMS* server. On the other hand, *PKI* operations are too complex for many applications, and using *XKMS* can ease the integration of *PKI* by moving the complexity of *PKI* operations to an *XKMS* server.

XML Key Management Specification (*XKMS*) contains two subprotocols: *XML* Key Information Service Specification (X-KISS) and *XML* Key Registration Service Specification (*X-KRSS*). *X-KISS* is used to locate and retrieve public keys from a

key server. It is to be used in encryption or signature verification. *X-KRSS* defines service interfaces for registration, revocation, and recovery of public keys from a key server.

The example in Table 3.5 shows that XKML works with document signatures. A client receives a signed *XML* document, and then sends the <ds:Keyinfo> element to the location service requesting that the <KeyName> and <KeyValue> elements be returned. In the <ds:Keyinfo> element it specifies a < ds:RetrievalMethod> for an X.509 certificate that contains the public key. The location service resolves the <ds:RetrievalMethod> to obtain an X.509v3 certificate. The certificate is parsed to obtain the public key value that is returned to the client. The < KeyName> returned is obtained from the certificate.

3.5 Languages and web services security

3.5.1 Extensible access control markup language

Extensible Access Control Markup Language(*XACML*) is an *XML* specification for expressing fine-grained information access policies in *XML* documents or any other electronic resources [6]. *XACML* expresses or communicates by using rules and policies. An access control mechanism is used to derive an access decision for a set of subjects and attributes. Access control lists in *XACML* are 4-tuples: subjects, target objects, permitted action and provision. The subjects include user IDs, groups or role names. The target objects allow granularity down to a single *XML* document element. The permitted action can be either read, write, create, or delete. A provision is an action for which a certain permission can be granted assuming that

```
Request:
  <Locate>
    <Query>
      <ds:KeyInfo>
        <ds:RetrievalMethod
          URI="http://www.PKeyDir.test/Certificates/01293122"
          Type="http://www.w3.org/2000/09/xmldsig#X509Data" />
      </ds:KeyInfo>
    </Query>
    <Respond>
      <string>KeyName</string>
      <string>KeyValue</string>
    </Respond>
  </Locate>
Response:
  <LocateResult>
    <Result>Success</Result>
      <Answer>
        <ds:KeyInfo>
          <ds:KeyName>
            O=XMLTrustCernter.org OU="Crypto" CN="Alice"
          </ds:KeyName>
          <ds:KeyValue>...</ds:KeyValue>
        </ds:KeyInfo>
      </Answer>
  </LocateResult>
```

Table 3.5: XKML validation request and respond

certain provisional actions are also executed. For example, in the access request, "Allow the school manager to create files in the student folder on the school server", the subject is the "school manager", the target resource is the "student folder on the school server", and the action is "create files". Using *XACML* can standardize the access control language in *XML*. It can make lower costs when people use it since

no writing policy in several languages and administators only need to understand one language.

Considering the following rule taken from the *XACML*, this example in Table 3.6 granta reading access to patient medical records only by their primary doctors. A patient has his/her patient records which include mental problem notes. The patient grants an access right to mental problem notes only to their primary care doctors. The primary care doctor then grants access to patient records to the associate doctor with access restrictions so that the associate doctor has no access to mental problem notes.

```
<content>
  <entry>
    <name> Alice</name>
    <record> mental problem </record>
  </entry>
</content>
<policy>
  <xacl>
    <object href="/contents"/>
    <rule>
      <subject>
        < uid primary care doctor />
      </subject>
      <action> name="read" permission="grant" </action>
    </rule>
  </xacl>
</police>
```

Table 3.6: XACML using example

3.5.2 Security assertion markup language

Security Assertion Markup Language (*SAML*) defines an *XML* framework for exchanging authentication and authorization information [7]. There are various *XML* security assertion, such as credentials, authentication, attribute and authorization. It also defines a Request/Response protocol definitions and *XML* protocol *SOAP* binding. *SAML* and *XACML* both share a lot of concepts and a domain – the domain of authentication, authorization, and access control. However, the problems they address in the same domain are different. *SAML* addresses authentication and provides a mechanism for transferring authentication and authorization decisions between cooperating entities, while *XACML* focuses on the mechanism for arriving at those authorization decisions. *SAML* can be used to share security information in Single Sign-on (*SSO*) with different systems and platforms. When using multiple networked systems a general requirement is "single sign-on", which means authenticating once and then sharing the result of authentication with multiple systems to avoid repeated authentication. For example, Logged-in (authenticated) users of Smith.com are allowed to access to their sister site Johns.com without relogin.

SAML is not concerned with confidentiality, integrity, or nonrepudiability of assertions in transit. The simplified authentication assertion example in Table 3.7 states that Smith (subject) was authenticated by "password" at a certain time towards Single Sign-on using.

3.5.3 Web services security

In April 2002, IBM and Microsoft issued a Web Services security architecture and roadmap that outline a stategy and specifications to bring different security technolo-

```
<Assertion>
 <AuthenticationStatement
    AuthenticationMethod="password"
    AuthenticationInstant="2003-12-05T10:00:00Z">
    <subject>
       <NameIdentifier
          SecurityDomain="Johns.com"
          Name="Smith"/>
       <ConfirmationMethod>
          http://...core-25/sender-vouches
       </ConfirmationMethod>
    </subject>
 </AuthenticationStatement>
</Assertion>
```

Table 3.7: SAML assertion example

gies together [5]. The WS-Security specification shows how *XML* Digital Signatures and *XML* Encryption may be used with *SOAP* [22] messages. Specifically, WS-Security describes enhancements to the existing *SOAP* message. It provides quality of protection for *SOAP* messages through applications in message integrity, confidentiality, and single message authentication. WS-Security provides technologies to build a wide variety of security models. WS-Security also provides a general-purpose mechanism for a*SSO*ciating security tokens with messages. Security tokens are designed to be extensible (e.g. support multiple security token formats) to accommodate a variety of authentication and authorization mechanisms. For example, a client might provide proof of identity and a signed claim to the bank where he/she has a fixed time-deposite certification. When receiving such a message in the Web service could then determine what kind of trust he/she places in the claim. An *XML* Signature may provide message integrity and security tokens can ensure that mes-

sages have originated from an appropriate sender and were not modified in transit. Similarly, *XML* Encryption may provide message confidentiality and security tokens can keep portions of a *SOAP* message confidential.

3.6 Conclusions

XML is today widely used in a large variety of applications and industry products as it has become the standard for describing data and documents circulated across the web. It is becoming a vital component of the emerging electronic business infrastructure. *XML* messages need to be secure to form the basis of business transactions. In this chapter we present a brief introduction to *XML* and Web services security standards and how they work together. *XML* Security standards define *XML* languages and processing rules for meeting common security requirements. These standards incorporate the use of the other *XML* Security standards, especially the core *XML* Digital Signature and *XML* Encryption standards. *SAML* and *XACML* are used for sharing policy statements.

Obviously, *XML* security standards will be essential as *XML* technologies are adopted for Web services. Security requirements have evolved and are required in many areas. These requirements mainly include the aspects of Authentication, Authorization, Integrity, Signature, Confidentiality, Privacy and Digital Rights Management. Understanding how *XML* can meet those requirements is important. However, the existing standards have established a good framework for developers who need to integrate security functionality into their *XML*-based applications.

Chapter 4

DTD Level authorization in XML document with usage control

This chapter presents a usage control model to protect information distributed on the web, which allows access restrictions directly at the DTD-level and *XML* document-level. It contains six different kinds of models built for *XML* DTD and *XML* documents. Then *XML* access control algorithms based on these models are given.

The information in this chapter is based on a published paper [117].

4.1 Introduction

An increasing amount of semi-structured data have become important to humans and programs in recent years. As *XML* has become a standard to exchange data, the need for the definition of a universal *XML* access control model to regulate access to *XML* data has become a necessity. This has led many researchers to propose different models which consider various issues for *XML* access control. Meanwhile, few solutions have been proposed to enforce these models properly, such as users having only access to what they are authorized.

In this chapter, we propose authorization models which adopt usage control to

manage access both at the instance-level and at the schema-level. Because in a traditional access control model given an access request, an algorithm computes a view of the target *XML* document based on the user's requirements right. It has analyzed authorization decisions on a subject's access to target resources before access is granted. In usage access control authorization decisions are not only checked and made before access, but are also repeatly checked during the access period. It may revoke the access permission according to the changes of the subject or object attributes. Meanwhile obligation and conditions become decision factors for the management of *XML* documents.

The remainder of this chapter is organized as follows: in section 2 we give an overview of basic *XML* access control models which have been proposed. The problems of these access control with *XML* documents are addressed. In section 3 we show our proposed authorization models for usage control. This includes *pre-Authorizations, ongoing-Authorizations, pre-Obligations, ongoing-Obligations*, *pre-Conditions*, *ongoing-Conditions* six models. The algorithm of *XML* access control is derived based on these six models. Section 4 is the conclusion.

4.2 *XML* access control models

Several authorization models have been proposed for regulating access to *XML* documents. Most of these models follow the well-established Discretionary Access Control (*DAC*) models [101]. In this section a simplified access control model for *XML* documents is introduced.

4.2.1 Basic models

The development of an access control system requires the definition of subjects and objects for which authorization rules must be specified and access controls must be enforced. Therefore, in the basic access controls, this takes the form of a 3-uple < *sign, subject, object* >. *Sign* denotes either a permission or a prohibition for the operation. *Subject* is the user or the group of users to which the rule applies. *Object* corresponds to elements or subelements in the *XML* documents. To expressive power of the access control model and the granularity of sharing, are directly supported by the XPath language [34]. In this thesis, a subset of XPath denoted by XP[], *, // is considered. This subset usually consists of node tests, the child axis(/), the descendant axis(//), wildcards(*) and predicates[...].

The *XML* access control models also implicit the cascading propagation of rules, which means that a rule propagates from an object to all its descendants in the *XML* hierarchy. Since the propagation mechanism and multiplicity of rules apply to a same user, a conflict resolution principle is required. Two policies are used for resolving conflicts in the models. There are *Denial-Takes-Precedence* and *Most-Specific-Object-Takes-Precedence*. It can be assumed that two rules *R1* and *R2* may conflict because either they are defined on the same object, or they are defined respectively on two different objects *O1* and *O2*, which are linked by an ancestor or descendant relationship. If a subject is granted access to an object, the path from the document root to this object is granted too. This *structural* rule keeps the document structure consistent with respect to the original one. Let us consider the bookstore *XML* document depicted in Figure 4.1.

The set of rules attached to a given subject on a given document is called an

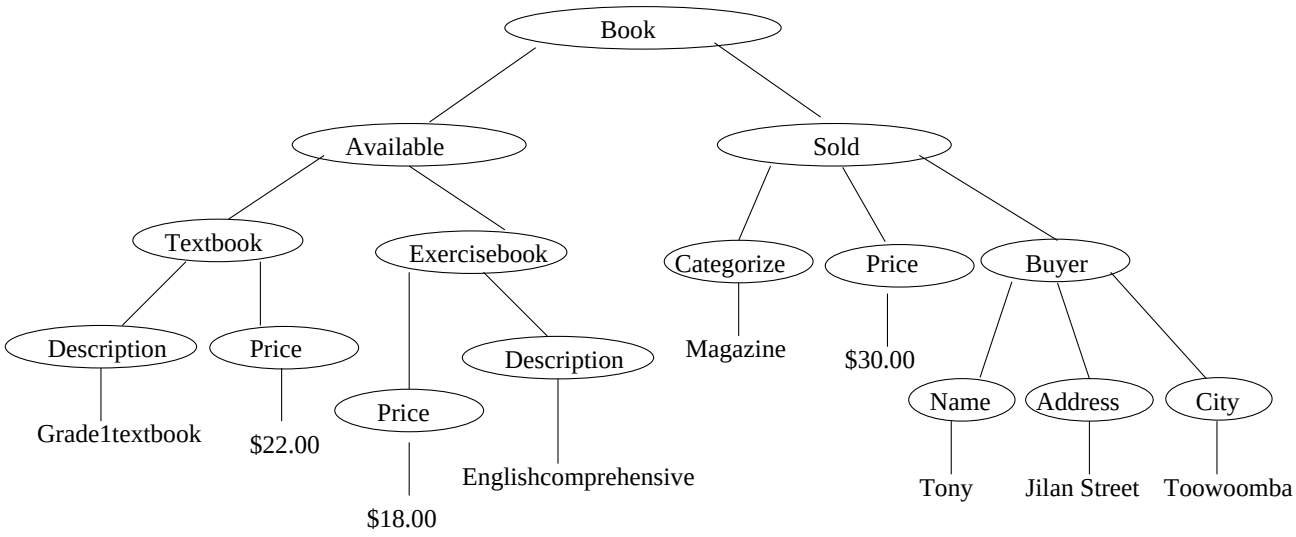

Figure 4.1: XML original document

access control policy. This policy defines an authorized view of this document and depending on the application context, this view may be queried. We consider that queries are expressed with the same XPath fragment as access rules.

An example is given in Table 2.1 for a bookstore document. The access control for the customer of selling information is defined as follows:

R1: <+, Smith, //books/available >

R2: <-, Smith, //books/sold >

This access control policy gives customer Smith the right to read information of all available books. The closed access control policy forbids his access to sold books. Let us consider the bookstore *XML* document depicted in Figure 4.1. Suppose that customer Tony has been given access to all available elements to create some statistics on the books of the bookstore (rule *R1*). The resulting authorized view is depicted in Figure 4.2.

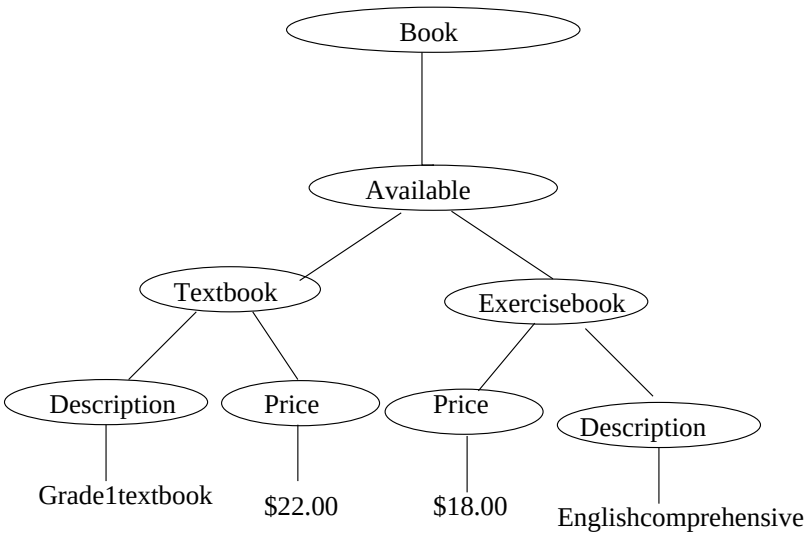

Figure 4.2: XML authorized view

4.2.2 Access control models and limits

Several approaches have been designed for the security of *XML* documents [37, 38, 73]. However, all those approaches have some limitations. Encryption and decryption skills are used in the protection of file levels that are not on a systematic level and their communications are between servers and clients rather than dissemination from clients. Traditional access control models focus on static authorization decisions based on subject permissions. However, these models present a similar approach in the work: a security administrator defines a set of policies at document level or at the Document Type Definition (DTD) level. Through access control, the system can restrict unauthorized users to access the resources in the system and guarantees the confidentiality and integrity of the resources. On the other hand, traditional authorization decisions are generated at the request's time but do not consider ongoing controls for long access or for revocation. Still, usage access control is a new access control model extending traditional access control models in multiple respects.

4.3 Authorization models with *XML* documents

In this section we consider authorization models for the DTD and *XML* documents adopting usage control. Based on the involvement of three decision factors: authorizations, obligations, and conditions, we develop models for usage control which consider enforcement. We assume that a usage request exists for an *XML* target object. Decision-making can be done either before (pre) or during (ongoing) the exercising of the requested right. Decision-making after the usage has no influence on the decision of current usage. Based on the requirements we have six possible cases as a model for usage control: pre-Authorizations, ongoing-Authorizations, pre-Obligations, ongoing-Obligations, pre-Conditions and ongoing-Conditions. Depending on the access requirements for the DTD and *XML* documents in the real world, it is possible to utilize more than one case. In this chapter, only the cases consisting of *Authorizations, Obligations* or *Conditions* along with *pre* or *ongoing* decisions are considered. Meanwhile here is a focus on developing the usage control models for the DTD and *XML* documents.

A. Usage control for pre-Authorization Model UCM_{preA}:

In a pre-Authorization usage control model, the decision process is performed before access is allowed. The following illustration of a usage decision that can be expressed on the documents DTD level and instance level are made in pre-authorizations using the *XML* DTD and *XML* documents in Table 2.1 and Table 2.2.

The UCM_{preA} model consists of the following components: $S, XDTD, XD, R, R_1$, $ATT(S)$, $ATT(XDTD)$, $ATT(XD)$ and usage decision Boolean functions $preA$,

$preA_1$ on $XDTD, XD$, respectively, where $S, XDTD, XD, R, R_1$ represent Subject, XML DTD, *XML* document and Rights required on XML DTD level and *XML* document, respectively. $ATT(S), ATT(XDTD), ATT(XD)$ represent attributes of subjects, *XML* DTD and *XML* document, respectively. $preA$ and $preA_1$ are predicates about authorization functions.

For example, consider the *XML* DTD and the *XML* document in Table 2.1 and Table 2.2:

$preA$ in DTD level is applicable to all bookstores.

D1: Information about the available books is publicly accessible for every reader.

D2: Information about the price of sold books is only accessible to administrative staff.

D3: The original price of books is not publicly accessible.

$preA_1$ in instance level is applicable to a bookstore in Toowoomba.

L1: The price of sold books is only accessible to members of financial staff.

L2: Smith can access information about books but only those books that have been discounted and sold in other cities.

1. $allowed(s, xdtd, r) \Rightarrow$

   $preA(ATT(s), ATT(xdtd), r)$,

   where $A \Rightarrow B$ means B is a necessary condition for A. In this example this predicate indicates that if subject s is allowed to access *XML* DTD level $xdtd$ with right r then the indicated condition $preA$ must match with D1, D2, D3.

2. $allowed(s, xd, r_1) \Rightarrow$

$preA_1(ATT(s), ATT(xd), r_1)$.

In this example the $allowed(s, xdl, r_1)$ predicate indicates that if subject s is allowed to access *XML* document xd with right r_1 then the decision function $preA_1$ must be matched with L1 and L2.

The UCM_{preA} model provides an authorization method on whether a subject can access the *XML* DTD level and Instance level document. The $allowed(s, xdtd, r)$ predicate shows that subject s can access information in the *XML* DTD level document. The $allowed(s, xd, r_1)$ predicate shows that subject s can access information in *XML* documents. In this process, private information in *XML* DTD and corresponding *XML* documents are restricted.

B. Usage control for ongoing-Authorizations Model UCM_{onA}:

A usage control model for ongoing-Authorizations model is used to check ongoing authorizations during access processes. In this model, usage requests are allowed without any 'pre' decision making.

The UCM_{onA} model has the following components: $S, XDTD, XD, R, R_1, ATT(S)$, $ATT(XDTD)$, $ATT(XD)$ as before, and ongoing usage decision functions onA on $XDTD$ (XML DTD level) and onA_1 on XD (XML document level). onA and onA_1 are used to check whether S can continue to access or not.

1. $allowed(s, xdtd, r) \Rightarrow true$,

   This is a prerequisite for ongoing authorization on $xdtd$.

2. $allowXMLed(s, xd, r_1) \Rightarrow true$,

   This is a prerequisite for ongoing authorization on xd.

3. $stopped(s, xdtd, r) \Leftarrow$

   $\neg onA(ATT(s), ATT(xdtd), r)$,

   The access of subject s to $xdtd$ is terminated if the ongoing authorization onA fails.

4. $stopped(s, xd, r_1) \Leftarrow$

   $\neg onA_1(ATT(s), ATT(xd), r_1)$.

   The access of subject s to xd is terminated if the ongoing authorization onA_1 fails.

In this model, usage decision Boolean functions are onA, onA_1 instead of preA, preA1. During this process the requested access is always allowed as there is no pre-authorization of all times. $allowed(s, xdtd, r)$ and $allowed(s, xd, r_1)$ are required to be *true*, otherwise ongoing authorization should not be initiated. Ongoing authorizations are active throughout the usage of the requested right, and some requirements are repeatedly checked for continued access. These checks are performed periodically based on time or event. In the process when attributes are changed and requirements are no longer satisfied, *stopped* procedures are performed. $stopped(s, xdtd, r)$ and $stopped(s, xd, r_1)$ indicate that rights r and r_1 of subject s on object *XML* DTD and *XML* document are revoked and the ongoing access is terminated. For example, if there is a limited number of simultaneous usage, only two administration staff can access information about the price of sold books in an object *XML* DTD level simultaneously. If a third administration staff requests access and passes the pre-authorization, the staff with the earlier time access is terminated. While this is a case of ongoing authorizations, it is important that the certificate should be evaluated in a *pre* decision.

C. Usage control for pre-Obligations Model UCM_{preB}:

UCM_{preB} introduces pre-obligations that have to be fulfilled before access is permitted. It will return true or false for usage decision depending on whether obligation actions have been fulfilled or not. This model consists of two steps: the first is to select required obligation elements for the requested usage, and then to evaluate whether the selected obligation elements have been fulfilled or not. Examples of pre-obligations require a reader to register by filling out forms before accessing online reading, and requiring a reader to click the ACCEPT box on a license agreement to read some books online. The pre-obligation action may be performed on a different object (e.g., register, license agreement) that the reader is trying to access (e.g., an e-book). This means that the pre-obligation action may be performed by some other subject. Hence obligation subjects, objects, and actions are added in the following UCM_{preB} model. The following model is described for *XML* DTD level and *XML* documents. It can be used for restricted information in *XML* DTD or in the *XML* documents.

The UCM_{preB} model has the following components: $S, XDTD, XD, R, ATT(S)$, $ATT(XDTD)$ and $ATT(XD)$ are as before; OBS, OBO and OB represent obligation subjects, objects, and actions, respectively; decision function $preObfilled$: $OBS \times OBO \times OB \rightarrow \{true, false\}$. As mentioned above, subject S and access object $XDTD$, XD may be different from OBS and OBO. The functions $preObfilled(s, xdtd, r)$ and $preObfilled(s, xd, r)$ are used to check if obligations are obeyed or not before the $subject(s)$ accesses the $object(xdtd, or, xd)$.

1. $allowed(s, xdtd, r) \Rightarrow preObfilled(s, xdtd, r)$.

   The $preObfilled(s, xdtd, r)$ function must be true if $subject(s)$ is allowed to

access $xdtd$ with right r.

2. $allowed(s, xd, r) \Rightarrow preObfilled(s, xd, r)$.

   The $preObfilled(s, xd, r)$ function must be true if s is allowed to access xd with right r.

This model indicates that obligations have to be fulfilled before s can access $xdtd$ or xd. Note that each obligation has to be true if there are more than two obligations.

D. Usage control for ongoing-Obligations Model UCM_{onB}:

Similar to pre-Obligations, Obligations are required to be fulfilled in UCM_{onB} models while rights are exercised. Ongoing-obligations may have to be fulfilled periodically or continuously. For example, when a reader accesses an e-book through the Internet within every 15 web pages, the reader may have to open an advertisement window. Alternatively, the reader may leave an advertisement window active all the time with inconvenience. The model is concerned with whether obligations have to be fulfilled.

The UCM_{onB} model has the following components: $S, XDTD, XD, R, ATT(S)$, $ATT(XDTD)$ and $ATT(XD)$ as before, OBS, OBO, and OB represent obligation subjects, objects, and actions, respectively; an ongoing decision function $onObfilled$: $OBS \times OBO \times OB \rightarrow \{true, false\}$. The ongoing functions $preObfilled(s, xdtd, r)$ and $preObfilled(s, xd, r)$ are used to check if obligations are continually obeyed or not during $subject(s)$ access $object(xdtd, or, xd)$.

1. $allowed(s, xdtd, r) \Rightarrow true$,

   A prerequisite for UCM_{onB}. This means that s is accessing XML DTD.

2. $allowed(s, xd, r) \Rightarrow true$,

   A prerequisite for UCM_{onB}. This means that s is accessing an *XML* document.

3. $stopped(s, xdtd, r) \Leftarrow \neg onObfilled(s, xdtd, r)$.

4. $stopped(s, xd, r) \Leftarrow \neg onObfilled(s, xd, r)$.

Where $stopped(s, xdtd, r)$ indicates that the access of s on $xdtd$ with r is revoked if the ongoing obligations fail. Alternatively, $stopped(s, xd, r)$ indicates that the access of s on xd with r is revoked if the ongoing obligations fail.

E. Usage control for pre-Conditions Model $UCM_(preC)$:

As described earlier, conditions define that certain restrictions have to be satisfied for usage. Conditions are not directly related to subjects and objects since they define environmental and system restrictions. Using conditions in the usage decision processes can provide finer-grained controls on usage. When focusing on this model on *XML* DTD and *XML* documents, the pre-conditions model has to be used before the requested rights are exercised. For example, suppose there are some requirements to restrict times for reading papers, such as papers can only be read 5 times, and printed 3 times. They then should be checked before a usage is allowed.

The UCM_{preC} model has the following components: $S, XDTD, XD, R, ATT(S)$, $ATT(XDTD)$ and $ATT(XD)$ as before, $preCON$(a set of pre-conditions), verify conditions function $preConSatisfied : preCON \rightarrow \{true, false\}$. The function $preConSatisfied$ is used to check whether the pre-conditions are satisfied or not.

1. $preC(s, xdtd, r) =$

   $\bigwedge_{preCon_i \in preCON} preConSatisfied(preCon_i)$,

   or

2. $preC(s, xd, r) =$

   $\bigwedge_{preCon_i \in preCON} preConSatisfied(preCon_i)$.

   All pre-conditions have to be checked if there are more than two conditions.

3. $allowed(s, xdtd, r) \Rightarrow preC(s, xdtd, r)$, or

4. $allowed(s, xd, r) \Rightarrow preC(s, xd, r)$.

Where $allowed(s, xdtd, r)$ and $allowed(s, xd, r)$ express that all conditions have to be satisfied before access is approved.

F. Usage control for ongoing-Conditions Model UCM_{onC}:

UCM_{onC} model requires conditions to be satisfied while rights are in active use. If violating any of the restrictions, the allowed right is revoked and the exercise is stopped. For example, *realOne player* does not work when the Windows XP system works on a safety module.

The UCM_{onC} model has the following components: $S, XDTD, XD, R, ATT(S)$, $ATT(XDTD)$ and $ATT(XD)$ as before, $onCON$(a set of ongoing conditions), verify ongoing conditions elements. $onConSatisfied : onCON \rightarrow \{true, false\}$, The function $onConSatisfied$ is used to check whether ongoing conditions are satisfied or not.

1. $onC(s, xdtd, r) =$

   $\bigwedge_{onCon_i \in onCON} onConSatisfied(onCon_i)$;

   or

2. $onC(s, xd, r) =$

   $\bigwedge_{onCon_i \in onCON} onConSatisfied(onCon_i)$;

or

All ongoing conditions are required to be checked.

3. $allowed(s, xdtd, r) \Rightarrow true$, or

4. $allowed(s, xd, r) \Rightarrow true$,

   A prerequisite for UCM_{onC}.

5. $stopped(s, xdtd, r) \Leftarrow \neg onC(s, xdtd, r)$.

6. $stopped(s, xd, r) \Leftarrow \neg onC(s, xd, r)$.

In practice, the above six models may need to be combined for access control. An authorization method for XML DTD and *XML* documents and their elements is obtained by checking users' (subjects') authorizations, obligations and conditions with continuity properties.

The following algorithm of *XML* access control is based on these models and introduces how to manage an *XML* document access control when a user (subject) applies for accesses to an *XML* document (target.xml) with right r. We obtain an authorization method for the *XML* document and its elements by checking users' (subjects') authorizations, obligations and conditions with continuity properties.

XML-based Algorithm:

Input: Access request: (u, r, target.xml)

Output: target'.xml

Method:

// Verify UCM_preA:

1) **if** $preA(ATT(s), ATT(xdtd), r) \bigcup preA(ATT(s), ATT(xd), r_1) = false$

// The process in pre-Authorization is not successful

2) ACCESS denied;

3) **endif**

// Verify UCM_onA:

4) **if** $preA(ATT(s), ATT(xdtd), r) \bigcup preA(ATT(s), ATT(xd), r_1) = false$

// The process in pre-Authorization has failed, do not need further verification.

5) Application denied;

6) **endif**

7) $onA(ATT(s), ATT(xdtd), r) \bigcup onA(ATT(s), ATT(xd), r_1) = false$

// The process in ongoing-Authorization is not successful

8) ACCESS stopped;

// Verify UCM_preB:

9) **if** $preObfill(s, xdtd, r) \bigcup preObfill(s, xdtd, r) = false$

// Obligations are not fulfilled and pre-Obligation is not passed.

10) ACCESS denied;

11) **endif**

//Verify UCM_onB:

12) **if** $allowed(s, xdtd, r) \bigcup allowed(s, xd, r) = false$

13) Stop verification.

14) **endif**

15) **if** $onObfill(s, xdtd, r) \bigcup onObfill(s, xd, r) = false$

// Obligations are not continually fulfilled and on-Obligation is not passed.

16) ACCESS is stopped;

17) **endif**

// Verify UCM_preC:

18) **if** $preC(s, xdtd, r) \bigcup preC(s, xd, r) = false$

// Conditions are not satisfied and pre-Condition verification is not passed.

19) ACCESS denied;

20) **endif**

//Verify UCM_onC:

21) **if** $allowed(s, xdtd, r) \bigcup allowed(s, xd, r) = false$

22) Stop verification.

23) **endif**

24) **if** $onC(s, xdtd, r) \bigcup onC(s, xd, r) = false$

// Conditions are not continually satisfied and on-Condition is not passed.

25) ACCESS is stopped;

26) **endif**

27) ACCESS target.xml is permitted;

28) Output target'.xml;

4.4 Conclusions

This chapter introduced the basis access control models and discussed access models for *XML* DTD and *XML* documents by using usage control. Usage control models provide an approach for the next generation of access control. We analyse not only

decision factors in usage control, such as authorizations, obligations and conditions, but also the continuity properties. This chapter has also illustrated six different kinds of models built for *XML* DTD and *XML* documents and *XML* access control algorithms based on these models. In addition, the work in this chapter has significantly extended previous work, such as the ongoing continuity for authorizations, obligations and conditions in usage control for *XML* DTD and *XML* documents. The methods presented in this chapter can be used to control *XML* documents in a dynamic environment. It also presents a new application for usage control.

Obviously, there is an increasing realization that traditional access control is not adequate for modern application needs. This chapter represents only a first step for DTD level authorization in *XML* documents with usage control, and much work is still to be done before these models can be used in practice.

Chapter 5

XML Schema in XML documents with usage control

The extensible markup language *XML* is a fundamental component in many *XML* web services. *XML* Schema provides a mechanism to build relationships between schemas and elements. In this chapter, a usage control model is presented which allows for access restrictions directly at schema-level and *XML* document-level. Finally, comparisons with related works are analysed.

The information in this chapter has been published in [118].

5.1 Some basic definitions

We identify two levels at which authorizations on *XML* documents can be defined, there are instance and the Document Type Definition (DTD) [79, 138, 117]. Instance level authorizations denote privileges that apply only to a specific document. DTD level authorizations specify the privileges of all documents following a given DTD. An *XML* Schema is an *XML*-based alternative to DTD. It defines the contents and relationships of elements in an *XML* instance. It supports complex constraints for *XML* components, such as elements, attributes, datatypes and groups. A well-

validated *XML* document must follow the format specified by one or several schemas. In the proposed access model, user access permissions are defined on schema or schema element level and will be transported to all *XML* instances specified by these schemas or elements. In the access control model the central authority uses *XML* schemas to specify the format of information to be changed. With the features of *XML* Schema, a flexible and easily-customized access control model can be achieved.

Access control has been considered as a major issue in the information security community since the beginning of the Information security discipline. Through access control, the system can restrict unauthorized users' access to the resources in the system and can guarantee the confidentiality and integrity of the resources [37, 91]. Traditional access control models primarily consider static authorization decisions based on the subjects' permissions on target objects. They focus on the protection of data in a closed environment. More recently research in authorization is about trust management. Trust management [18] relates authorization to a user's capability and properties. These access models have used the control of access to server-side objects. Digital rights management (*DRM*) [38] is used for objects disseminated. Current *DRM* solutions are largely focused on payment-based dissemination controls. Because each access control is different, the authorization decisions are generated at request time but do not consider ongoing controls for long access or for revocation. We need a comprehensive, systematic approach for controls on usage of digital objects [95, 106]. Usage control is an access control model which extends traditional access control models. It protects information distributed on the Web, which allows for access restrictions directly to structures and documents.

In this chapter, we propose authorization models which adopt usage control to manage access both at the instance-level and at the schema-level. Traditional access

control gives an access request and an algorithm which computes a view of the target *XML* document based on the user's required right. It analyses authorization decisions on a subject's access to target resources before access. However, usage access control authorization decisions are not only checked and made before access, but are also repeatly checked during the access period. Meanwhile obligations and conditions become decision factors for the management of *XML* documents.

Based on the involvement of three decision factors: authorizations, obligations, and conditions, the focus will turn to developing the usage control models for the *XML* Schemas and *XML* documents. Similar to Chapter 4, there are six possible cases as a model for usage control: pre-Authorizations, ongoing-Authorizations, pre-Obligations, ongoing-Obligations, pre-Conditions, ongoing-Conditions.

The following section reviews the differences between this work and those of others. Section 3 outlines proposed authorization models for usage control using *XML* Schema. It includes pre-Authorizations, ongoing-Authorizations, pre-Obligations, ongoing-Obligations, pre-Conditions and ongoing-Conditions models. Section 4 concludes the chapter.

5.2 Related work

Xinwen, Jaehong and Ravi [142] introduced an extended RBAC model for *XML* security . In their model, permissions are defined based on *XML* Schema components and will be transported to all instances. The permission reuse through these hierarchies provides the security administration. Several constraints are presented in the model. The proposed model can be modularly deployed and flexibly administrated in distributed environments. Their model can be readily applied to no-schema based *XML* instances and instance level authorizations. However, this work substantially

differs from their proposal. The main differences in this approach are in the following aspects. First, their protocol is based on role based access control (RBAC) and hence it focuses on permissions-role assignments, object hierarchies and constraints. This approach is based on usage control, and the characteristics of various access authorizations are analyzed, and detailed models for different kinds of authorizations are presented. Second, their approach does not mention how to update users' permissions on *XML* objects when their conditions or obligations have changed. This is an important state for *XML* documents on the Internet since users always alter their conditions or obligations. By contrast, users in this study's scheme have to pass pre-Authorizations and ongoing-Authorizations as well as pre-Obligations, pre-Conditions and ongoing-Obligations and ongoing-Conditions. This indicates that this method will be much more powerful in dynamic environments.

Elisa and Elena [12] presented an access control system supporting selective distribution of *XML* documents among possible large user communities by using a range of key distribution methods. In their papers, a formal model of access control policies for *XML* documents is given. It focuses on key distribution methods to protect *XML* documents. The approach consists of encrypting different portions of the same document according to different encryption keys, and selectively distributing these keys to the various users. By contrast, this work provides a rich variety of options that can deal with *XML* documents. Users can access *XML* documents with their keys at any time, even when their properties are updated. This means there is no ongoing authorization for users. In this scheme, users have to satisfy pre-Authorizations, pre-Obligations and pre-Conditions as well as ongoing-Authorizations, ongoing-Obligations, ongoing-Conditions.

5.3 Authorization models with *XML* schema

In a real environment, the resource of *XML* objects is based on different *XML* Schemas from various servers and organizations. Therefore there will be vast number of schema components. On the other hand, the vast number of users will make subjects complex. With all these features, the security administration will be very complex in both centralized and decentralized deployments.

An *XML* provides a uniform mechanism to solve problems in the environment. To apply the modularity, and extensibility information flow in an *XML* format, all the components in the models, such as subjects, subject attributes, objects, object attributes, rights, authorizations, obligations, and conditions will be specified in an *XML* format. Through some *XML* operations, the content of a target *XML* document will be customized for users to access with their permissions. Particular schemas will identify security related information, and *XML* instances can be centrally stored or distributed among the organizations. Figure 5.1 shows the implementation layout of the *XML* architecture based on the proposed *XML* document (see Table 2.1) and its Schema (see Table 2.3), as presented in Chapter 2. The entire messages transported among the services are identified in *XML*. *XML* requests and responses are *XML* messages, whose formats will be defined in schemas.

In this model, there are schema objects and instance objects. Schema objects (SO) are mapped to instance objects (IO) by an instance mapping (IM) function. In this authorization model, a Schema object is an *XML* Schema or schema component(s). An instance object is an *XML* instance or instance component(s). Instance Mapping is a mapping between SO and IO. In the following the details of the model

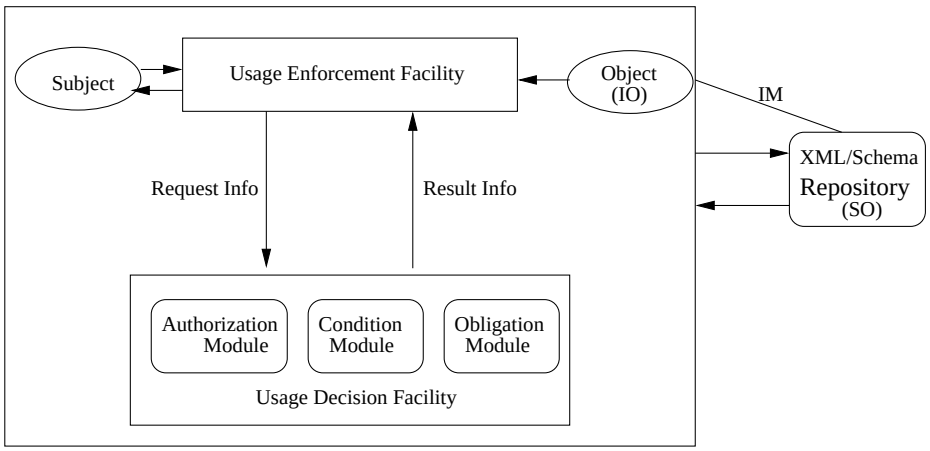

Figure 5.1: XML architecture

process are explained.

A. Usage control for pre-Authorization Model(UCM_{preA})

In a pre-Authorization usage control model, the decision process is performed before access is allowed. The following is an illustration of a usage decision that can be expressed on the document's *XML* Schema and instance level, which are made in pre-authorizations.

The UCM_{preA} model consists of the following components: $S, XSch, XD, R, R_1$, $ATT(S)$, $ATT(XSch)$, $ATT(XD)$, IM and usage decision Boolean functions $preA$, $preA_1$ on XS, XD, respectively, where $S, XSch, XD, R, R_1$ represent Subject, *XML* Schema object, *XML* document object and Rights required on *XML* Schema level and *XML* document, respectively. $ATT(S), ATT(XSch), ATT(XD)$ represent attributes of subjects, *XML* Schema object and *XML* document object, respectively. IM represent Instance Mapping between SO and IO. $preA$ and $preA_1$ are predicates about authorization functions.

1. $allowed(s, xsch, r) \Rightarrow$

$preA(ATT(s), ATT(xsch), r)$,

where $A \Rightarrow B$ means B is a necessary condition for A. In this example this predicate indicates that if subject s is allowed to access XML Schema level $xsch$ with right r then the indicated condition $preA$ must be true.

2. $allowed(s, xd, r_1) \Rightarrow$

   $preA_1(ATT(s), ATT(xd), r_1)$.

   In this example the $allowed(s, xd, r_1)$ predicate indicates that if subject s is allowed to access XML document xd with right r_1 then the decision function $preA_1$ must be true.

3. $IM : SO \rightarrow 2^{IO}$.

   where IM is a one-to-many mapping relationship from SO to IO. In this model, we use IM to implicitly specify the authorization in the instance level. Specifically, the attributes defined on schema objects will be transported to all its instance objects.

The UCM_{preA} model provides an authorization method on whether a subject can access the XML Schema and Instance level document. The $allowed(s, xsch, r)$ predicate shows that subject s can access the XML Schema object. The $allowed(s, xd, r_1)$ predicate shows that subject s can access the XML document object. In this process, private information in XML Schema and corresponding XML documents are restricted.

B. Usage control for ongoing-Authorizations Model (UCM_{onA})

A usage control model for an ongoing-Authorizations model is used to check ongoing authorizations during access processes. In this model, usage requests are

allowed without any 'pre' decision making. The process of UCM_{onA} with Schema is similar to the DTD level authorization with an *XML* document that was presented in Chapter 4. During this process both *allow(s,xsch, r)* and *allow(s,xd, r1)* are required to be *true*, otherwise ongoing authorization should not be initiated. *stopped(s,xsch, r)* and *stopped(s,xd, r1)* indicate that right r and *r1* of subject s on object *XML* Schema and *XML* documents are revoked and ongoing access terminated.

The UCM_{onA} model has the following components: $S, XS, XD, R, R_1, ATT(S)$, $ATT(XSch)$, $ATT(XD)$ and IM as before, and ongoing usage decision functions onA on $XSch$ (*XML* Schema level) and onA_1 on XD (*XML* document). onA and onA_1 are used to check whether S can continue to access or not.

1. $allowed(s, xsch, r) \Rightarrow true$,

   This is a prerequisite for ongoing authorization on *XML* Schema object $xsch$.

2. $allowed(s, xd, r_1) \Rightarrow true$,

   This is a prerequisite for ongoing authorization on *XML* document object xd.

3. $stopped(s, xsch, r) \Leftarrow$

   $\neg onA(ATT(s), ATT(xsch), r)$,

   The access of subject s to xs is terminated if the ongoing authorization onA fails.

4. $stopped(s, xd, r_1) \Leftarrow$

   $\neg onA_1(ATT(s), ATT(xd), r_1)$.

   The access of subject s to xd is terminated if the ongoing authorization onA_1 is failed.

5. $IM : SO \rightarrow 2^{IO}$.

   The mapping relationship is the same as that in the pre-Authorization model.

The process of other authorization models, such as pre-Obligations, ongoing-Obligations, pre-Conditions and ongoing-Conditions are omitted.

The following algorithm is based on these models and introduces how to manage an *XML* document access control when a user (subject) applies to access an *XML* Schema with right r and an *XML* document with right r_1. The output of an access control decision is required to satisfy some expected schemas. The function $im(target'.xsd)$ is to check if output $target'.xml$ can be validated by target'.xsd. Since the authorization process can remove some parts of the input object, the output may not satisfy some particular schema, which is required by most applications. In this case, the access will be denied.

XML-based Algorithm:

Input: Access request: (u, r, target.xml)

Schema of target: target.xsd

Schema of expected output: target'.xsd

Output: target'.xml

Method:

// Verify UCM_preA:

1) **if** $preA(ATT(s), ATT(xsch), r) \bigcup preA(ATT(s), ATT(xd), r_1) = false$

// The process in pre-Authorization is not successful

2) ACCESS denied;

3) **endif**

4) **if** $target'.xml < im(target'.xsd)$

5) Output $target'.xml$;

6) **else** ACCESS denied;

7) **endif**

// Verify UCM_onA:

8) **if** $preA(ATT(s), ATT(xsch), r) \bigcup preA(ATT(s), ATT(xd), r_1) = false$

// The process in pre-Authorization fails, no need for further verification.

9) Application denied;

10) **endif**

11) $onA(ATT(s), ATT(xsch), r) \bigcup onA(ATT(s), ATT(xd), r_1) = false$

// The process in ongoing-Authorization is not successful

12) ACCESS stopped;

13) **if** $target'.xml < im(target'.xsd)$

14) Output $target'.xml$;

15) **else** ACCESS denied;

16) **endif**

5.4 Conclusions

In this chapter some basic definitions of *XML* and *XML* Schema are introduced, and access models for *XML* Schemas and *XML* documents by using usage control are discussed. Compared to *XML* DTD, *XML* Schema has richer potential usage than *XML* DTD. This chapter also illustrates two different kinds of models built for *XML* Schemas and *XML* documents. Access control in Web services will be popular in the future since all messages and protocols in Web services are in *XML* format. Only

using traditional access control is not adequate for modern application needs. The Schema based usage control model can be a good solution for Web service security.

Chapter 6

Using usage control to access XML databases

The *XML* database is becoming increasingly important since it consists of *XML* documents. Several applications for supporting a selective access to data are available over the web. Usage access control has been proven to be efficient in improving security administration with flexible authorization management. Objects-oriented database systems represent complex data structures and *XML* databases may be stored in objects-oriented database systems. Therefore authorization models for *XML* databases are similar to the models for object-oriented databases. In this chapter, we propose usage control models to access *XML* databases and compare them with a methodology designed for object-oriented databases. Comparisons with related work are analysed. We have analysed the characteristics of various access authorizations and presented detailed models for different kinds of authorizations.

This chapter is organized as follows: Section 1 illustrates the background of *XML* databases. *XML* and databases are reviewed in this section. Section 2 has a view of the Objects-oriented database systems (OODB) authorization model. Section 3 reviews the differences between this work and others. Section 4 shows the proposed authorization models for usage control using *XML* databases. The models include

pre-Authorizations, ongoing-Authorizations, pre-Obligations, ongoing-Obligations, pre-Conditions and ongoing-Conditions with *XML* databases. Finally, the conclusions are in Section 5.

Most information in this chapter has been accepted by an international journal in [119].

6.1 *XML* and databases

An *XML* document is a collection of data. It is a self-describable, exchangeable and a tree graphic structure description data set. *XML* documents fall into two categories: $data-centric$ and $document-centric$ [38]. Data-centric documents are those where *XML* is used as a data transport. For example, dynamic Web pages are a special case of data-centric documents. Document-centric documents are designed for human reading. Examples are books, emails and advertisements. They are characterized by irregular structures and mixed contents.

To store and retrieve data in data-centric documents, you need to know how well structured your data is. For highly structured data, you will use an *XML*-enabled database for data storage, such as a relational or object-oriented database, and some sort of data transfer software such as middleware [21]. If your data is semi-structured, you may have two choices. You can fit your data into a well-structured database, such as a relational database, or you can store it in a native *XML* database. The native XML database is specialized for storing *XML* data and stores all components of the *XML* model intact [37]. To store and retrieve document-centric documents, you will need a native XML database. Some native XML database models are stored in the relational and object-oriented databases. For example, in the relational database storage Document Object Model (DOM),

there are elements, attributes, entity, and other entities cited forms. As traditional databases add native *XML* capabilities and native *XML* databases support the storage of document fragments in external (usually relational) databases, the access control models for traditional databases, such as relational or object-oriented databases and native XML databases, could be used in the same way.

A recent development in the database field has been the introduction of semi-structured and self-describing data, a collection of data in *XML* format called XML Databases [130]. Some work [99, 130] on the relationship between securing *XML* documents and object oriented databases (OODB) has been done. However, there appears to be no detailed discussion of how the usage access model can be applied to XML databases. In this chapter, authorization models are proposed which adopt usage control to manage access to *XML* based databases.

An example of an *XML* document containing information about company staff is shown in Table 6.1.

As one of the main validation specification mechanisms [19, 96] XML Schema can be attached to *XML* documents, specifying the rules that *XML* documents may follow. XML Schemas are extensible to future additions. *XML* Schemas provide a means for defining the structures, content and semantics of *XML* documents. The example in Table 6.1 displays an *XML* Schema for a corresponding valid *XML* instance in Table 6.2.

6.2 The OODB authorization model

Object-oriented database systems (OODB) [98] are an important emerging technology for applications in business, industry, and many other areas. OODB is the most

```
<?xml version="1.0" encoding="UTF-8? ">
<StaffInfo xmlns="http://www.company.com/StaffInfo">
  <company StaffId="12345">
    <GeneralInfo>
      <name > Tony Mahanee </name>
      <address > 1, Smart Street </address>
      <email > Tony@hotmail.com </email>
    </GeneralInfo>
    <WorkInfo>
      <workarea>implement</workarea>
      <developarea>research</developarea>
    </WorkInfo>
    <FinancialInfo> $3800.00 </FinancialInfo>
  </staff>
</staffInfo>
```

Table 6.1: XML document example for staff information

popular data model to represent complex data structures [71]. An *XML* database is a standard for representing semi-structured data. Schemas of XML databases can represent dynamic data structures, such as lists, trees and graphs. Since OODB and *XML* databases are suitable to represent complex objects, they may have the same authorization models. Therefore, the OODB authorization model can be applied to *XML* Schema and documents [142]. The OODB authorization model presented by Rabitti et al [98] is a discretionary access control model for an object-oriented database. It models an authorization as a triple:

$$f : S \times O \times A \rightarrow (True, False)$$

Where S represents the set of subjects, O represents the set of objects and A is the possible authorization types (access modes) in a system. The models of authorization supported in existing database systems are all designed for relational,

```
<?xml version="1.0" encoding="UTF-8"? >
 <xs:schema>
  targetNamespace="http://www.company.com/StaffInfo"
  xmlns:xs= "http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
   <xs:annotation>
    <xs:documentation >
     <Staff Information Instance>
    </xs:documentation >
   </xs:annotation >
  <xs:element name="StaffInfo">
   <xs:sequence>
    <xs:element name="staff" type="xs:string" />
      <xs:complexType name="GeneralInfo">
       <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="address" type="xs:string" />
        <xs:element name="email" type="xs:string" />
       </xs:sequence>
      </xs:complexType>
      <xs:complexType name="WorkInfo">
       <xs:sequence>
        <xs:element name="workarea" type="xs:string" />
        <xs:element name="developarea" type="xs:string" />
       </xs:sequence>
      </xs:complexType>
      <xs:element name="FinancialInfo" type="xs:string" />
      <xs:attribute name="StaffInfo" type="xs:string" />
   </xs:sequence>
 </xs:schema>
```

Table 6.2: XML schema example for staff information

hierarchical, or network models of data.

The basic idea of an access control model is to group subjects into access control groups and to grant authorizations in terms of access types, such as *read*, *write*, and *delete* [99]. These access types are usually ordered such that the authorization for one right may include others. Thus authorization for a *delete* may imply authorization for a *write*, which in turn may imply authorization for a *read*. Database systems usually define authorizations for the schema entities, such as classes, attributes, and indexes [99] . In the database object part of the authorization, Rabitti et al., discusses two graphs: the authorization object schema (AOS) and the authorization object graph (AOG). The Figures 6.1 and 6.2 show the examples of AOS and AOG, respectively. The edges in the AOG represent relationships between objects. The nodes in both the AOS and AOG deal with collections of objects of a given type depending on how OODB handles sets of objects. AOS looks at the possible granules defined by the schema for OODB; AOG considers actual object instances on the database. All access control problems ultimately point at a fundamental question: subjects allowed to access of type on object o. The answer to any access control request can now be obtained by utilizing a function f that determines if the corresponding authorization (s, o, a) is true or false.

6.3 Related work

Jingzhu and Sylvia [130] introduced a role based approach to access control for an *XML* database. In their model, they provide a general access control methodology for parts of *XML* documents, combining role based access control as found in the Role Graph Model, with a methodology originally designed for object-oriented databases. Several constraints are included in the model. Their protocol is based

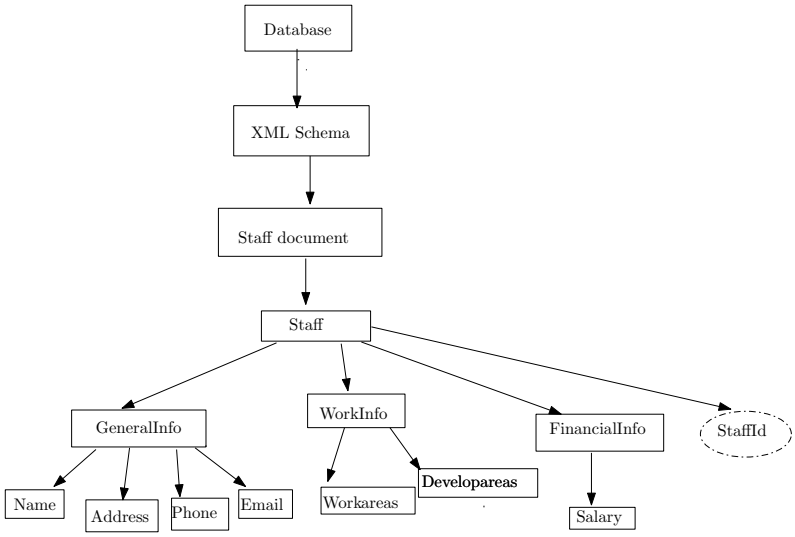

Figure 6.1: Authorization object schema

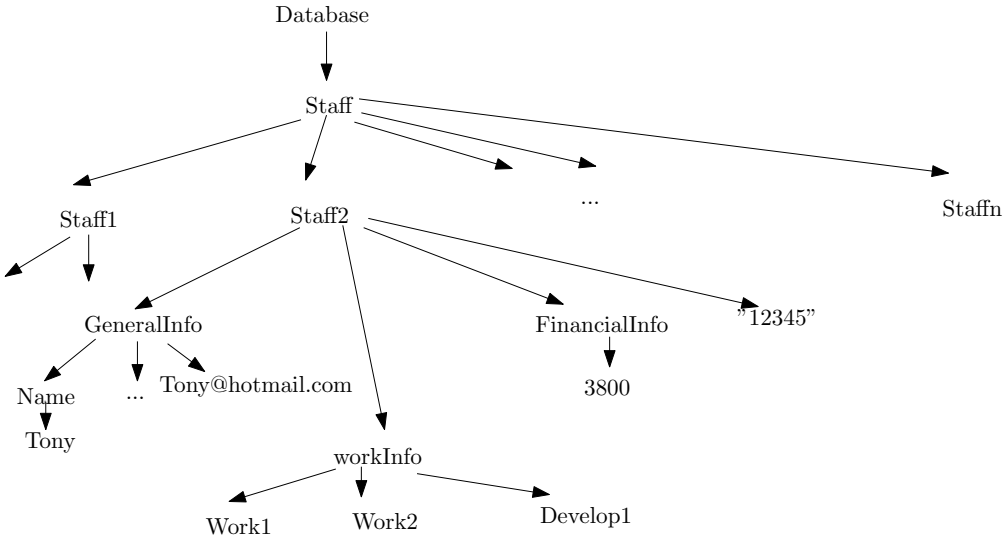

Figure 6.2: Authorization object graph

on RBAC and hence it focuses on permissions-role assignment, objects hierarchies and constraints. By contrast, this approach is based on usage access control; the characteristics of various access authorizations have been analysed and six detailed models for different kinds of authorizations have been presented. This is the main feature of this access control model which is not found in any access control model previously proposed for *XML* databases since users always alter their conditions or obligations.

Alban, et al [59] presented an access control model for regulating access to *XML* documents. In their papers, they use the XPath language to address *XML* fragments and the *XSLT* language to compute the view. The model offers the possibility of defining content-based authorization rule. By contrast, this work provides a rich variety of options that can deal with *XML* database documents. In this scheme, users have to satisfy pre-Authorizations, pre-Obligations, pre-Conditions ongoing-Authorizations, ongoing-Obligations and ongoing-Conditions whilst accessing *XML* databases.

6.4 Usage control models with *XML* databases

6.4.1 The objects

An *XML* schema defines *XML* documents with a hierarchical structure which contains attributes and elements. Elements can have sub-elements nested in any level. In this chapter we assume that the objects using access control consist of documents which conform to an *XML* Schema. For the Rabitti model [98] the *XML* Schema can be used to construct the AOS, and the *XML* documents make up the AOG. An AOS and AOG, for the example in Section 2, are shown in Figures 6.1 and 6.2, respectively. In the AOS in Figure 6.1, we have indicated elements by rectangles, at-

tributes by ovals. So *Staff*, *GeneralInfo*, *WorkInfo*, and *FinancialInfo* are elements, *StaffId* is an attribute of element Staff. In the AOG of Figure 6.2 , the element values are shown as their string value and attribute values are in quotes. *Tony*, *Tony@hotmail.com* and *3800* are string values of elements and *12345* is the value of an attribute.

6.4.2 Usage control models with *XML* databases

For the XML databases with the usage control model, the concepts of subjects and objects are similar to OODB authorization models. A right represents the access of a subject to an object, such as *read* or *write*. Operations on parts or all of an *XML* document, such as reading, changing the contents (update), adding sub-elements or attributes (extend) and following a pointer and applying *XLS* transformations are possible. This is similar to component authorization types in a discretionary access control model for an object-oriented database.

Authorization models for *XML* database adopting usage control is discussed next in this section. *XML* documents provide the information structure and semantics in a web environment and *XML* documents will be stored in a relational, object-oriented database or *XML* Native database. The authorization models for an *XML* Native database model are the same as the OODB model. Based on the three usage control components, Authorization, Obligations and Conditions, six cases are developed as core models: pre-Authorization, ongoing-Authorization, pre-Obligations, ongoing-Obligations, pre-Conditions, ongoing-Conditions using *XML* database.

1. pre-Authorization Model (UCMpreA) with *XML* databases

   The *UCMpreA* model provides an authorization method on whether a sub-

ject can access the *XML* database. It consists of the following components: $S, XO, R, ATT(S)$, $ATT(XO)$ and usage decision functions $preA$, respectively. S, XO, R, represent Subject, XML object, Rights required on XML object access modes, respectively. In addition, $ATT(S), ATT(XO)$, represent attributes of a subject and an attribute of an XML object, respectively. In usage control, the authorization decision is made based on subject attributes and object attributes. In the $UCMpreA$ model the decision process is:

$allowed(s, xo, r) \Rightarrow$

$preA(ATT(s), ATT(xo), r)$.

This predicate indicates that if subject s is allowed to access XML object xo with right r, then the indicated condition $preA$ must be true. This corresponds roughly to the discretionary access control model. $preA$ is very similar as the triple relationship f in the OODB authorization model mentioned in the previous section. The three components $ATT(s)$, $ATT(xo)$ and r replace S, O, A in the OODB authorization model. For example, in applying $UCMpreA$ to Figures 6.1 and 6.2, for the element staff $Tony$, his financial records can only be accessed by himself and financial administrators. Before they want to access this information, they have to provide a username (subject) and a password (subject attribute). In the meantime they need to know $Tony$'s (object) staffId (object attribute). Then they are able to read (right) Tony's financial records.

2. ongoing-Authorizations Model (UCMonA) with *XML* databases

   With the *XML* database, the $UCMonA$ model has the same components: S, XO, XD, R, $ATT(S), ATT(XO)$ as $UCMpreA$. The ongoing usage decision functions onA is used to check whether S can continue to access or not. It includes two processes:

$allowed(s, xo, r) \Rightarrow true$,

$stopped(s, xo, r) \Leftarrow onA(ATT(s), ATT(xo), r)$.

The $allowed(s, xo, r)$ is a prerequisite for ongoing authorization on XML object xo. Compared to the OODB authorization model, onA is similar to the triple relationship f. $ATT(s)$, $ATT(xo)$, r components also replace S, O, A in the OODB authorization model. The access of subject s to xo is terminated if the ongoing authorization onA fails.

3. pre-Obligations Model (UCMpreB) with *XML* databases

   $UCMpreB$ introduces pre-obligations that have to be fulfilled before access is permitted. For example, an administrator is required to register by filling forms before accessing one staff member's financial information. When using the $UCMpreB$ model to access *XML* database documents, the $UCMpreB$ model has the same components: $S, XO, R, ATT(S), ATT(XO)$ as $UCMpreA$, and a decision function $preObfilled : OBS \times OBO \times OB \rightarrow (true, false)$. OBS, OBO and OB represent obligation subjects, objects, and actions, respectively; the decision function $preObfilled$ checks if obligations are obeyed or not before the $subject(s)$ accesses the $object(xo)$. $preObfilled : OBS \times OBO \times OB \rightarrow (true, false)$ is the same as the triple relationship f: $S \times O \times A \rightarrow (true, false)$ in the OODB authorization model. The $preObfilled$ function must be true if subject(s) is allowed to access XML object xo with right r.

4. ongoing-Obligations Model (UCMonB) with *XML* databases

   Different from the pre-Obligations model, the Ongoing-obligations model may have to be fulfilled periodically or continuously. For example, when an administrator accesses the financial information through the Internet for 15

continuous days, she/he may have to input a password repeatedly. Using the $UCMonB$ model with XML databases involves the same components: $S, XO, R, ATT(S)$ and $ATT(XO)$ as $UCMpreA$ and an ongoing decision function $onObfilled : OBS \times OBO \times OB \rightarrow (true, false)$. OBS, OBO, and OB represent obligation subjects, objects, and actions, respectively. The ongoing function $onObfilled$ is used to check if obligations are continually obeyed or not while subject(s) is accessing object (xo) in an XML database document. $onObfilled : OBS \times OBXMLO \times OB \rightarrow (true, false)$ is the same as the triple relationship $f : S \times O \times A \times \rightarrow (true, false)$ in the OODB authorization model.

5. pre-Conditions Model (UCMpreC) with XML databases

   Usually the pre-conditions model has to be used before requested rights are used. An example of pre-conditions is a time restriction for accessing information. They should be checked before a usage is allowed. The $UCMpreC$ model with XML databases has the same components: $S, XO, R, ATT(S)$, and $ATT(XO)$ as $UCMpreA$ and $preCON$ (a set of pre-conditions) is for verifying conditions, $preCON \rightarrow (true, false)$. The function $preConSatisfied : S \times O \times R \rightarrow 2^{preCon}$ is used to check whether the pre-conditions are satisfied or not. It has two processes during the access:

   $allowed(s, xo, r) \Rightarrow$

   $preC(s, xo, r)$.

   $preC$ is very similar to the triple relationship f in the OODB authorization model. s, xo, r also replace S, O, A in the OODB authorization model. $allowed(s, xo, r)$ expresses that all conditions have to be satisfied before access is approved.

6. ongoing-Conditions Model (UCMonC) with *XML* databases

   $UCMonC$ model requires conditions to be satisfied while rights are in active use. If any of the conditions is not satisfied, the allowed right is revoked and the exercise is stopped. For example, if the staff information system status changes to special modes, accesses for certain users may be terminated.

   For the usage access control with *XML* databases the six models all include some functions. These functions, such as $preA, onA, preObfilled$, $onObfilled$, $preC$, etc are very similar to $f : (S \times O \times A)$ in the OODB authorization model. However, the usage authorization method for *XML* database focuses on checking users' (subjects') authorizations, obligations and conditions with continuity properties. It can also be used for different processes. In practice, two or more of the six models of pre-Authorizations, ongoing-Authorizations, pre-Obligations, ongoing-Obligations, pre-Conditions and ongoing-Conditions may need to be combined for access control.

6.5 Conclusions

In this chapter we review *XML* databases and the OODB authorization model. We discuss access control models for *XML* databases by using usage control. OODB authorization control is used in database system access control. It only has an authorization component, and it does not include obligation and condition components. Usage control encompasses traditional access control, trust management and provide an approach for the next generation of access control. It covers both security and privacy issues of current business and information systems. Compared to the OODB authorization model, the usage control model for accessing *XML* databases has a wide scope of applications. In this chapter, a foundation for further research and

development on the usage control model with *XML* databases has been provided.

Chapter 7

Authorization algorithms for permission-role assignments

Permission-role assignment (PRA) is one important process in Role-based access control (RBAC) which has been proven to be a flexible and useful access model for information sharing in distributed collaborative environments. However, problems may arise during the procedures of PRA. Conflicting permissions may assign to one role, and as a result, the role with the permissions can derive unexpected access capabilities.

This chapter aims to analyze the problems during the procedures of permission-role assignments in distributed collaborative environments and to develop authorization allocation algorithms to address the problems within permission-role assignments. The algorithms are extended to the case of PRA with the mobility of permission-role relationship. We do not specify *XML* documents in this chapter since the results in the chapter can also be applied to other documents in web service systems.

7.1 Introduction

Authorization is crucial for providing fine-grained access control to data in *XML* document. Authorizations use path expressions of XPath for locating data in documents, its definition is related to structure of the document. However, the structure of *XML* documents tends to change by various reasons such as application extension and information exchange between organizations. Therefore, authorizations must be revised whenever they become incompatible with a new structure of the document [29]. *XML* is going to be used in many environments such as application integration and Web Services due to its platform-independent feature. Security of *XML* documents is a basic problem, especially in enterprise with large number of users and *XML* objects as well as complex authorizations administration [28]. RBAC has been proven to be efficient to improve security administration with flexible authorization management. As far as we know, no previous work has discussed the problem of authorizations algorithms for permission-role assignments in RBAC [130]. Authorization algorithms of permission-role assignments in RBAC for a target DTD instance are proposed in this chapter.

The National Institute of Standards and Technology developed the role-based access control (RBAC) prototype [50] and published a formal model [51]. RBAC has been widely used in database system management and distributed environments since it enables managing and enforcing security in large-scale and enterprise-wide systems. RBAC involves individual users being associated with roles as well as roles being associated with permissions (Each permission is a pair of objects and operations). As such, a role is used to associate users and permissions. A user in this model is a human being. A role is a job function or job title within the organization associated with authority and responsibility.

Permission is an approval of a particular operation to be performed on one or more objects. As shown in Figure 7.1, the relationships between users and roles, and between roles and permissions are many-to-many. (i.e. a permission can be associated with one or more roles, and a role can be associated with one or more permissions).

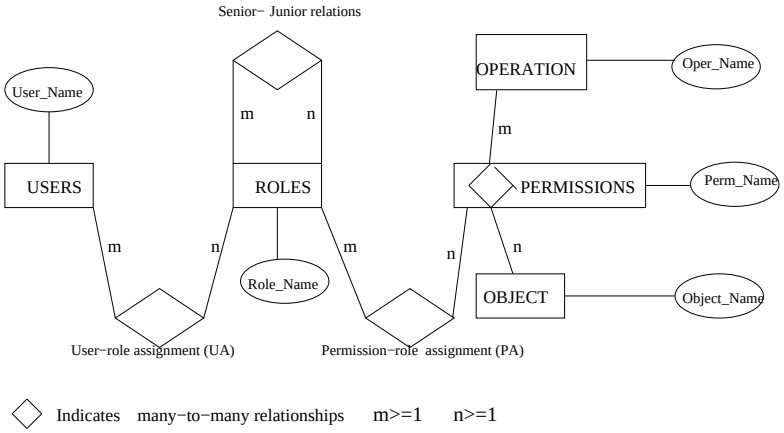

Figure 7.1: RBAC relationship.

Recently, role based access control (RBAC) has become widely used in database system management and operating system products because of its management advantages [133]. In 1993, the National Institute of Standards and Technology (NIST) developed prototype implementations, sponsored external research [50], and published formal RBAC models [51, 65]. Many organizations prefer to centrally control and maintain access rights, not so much at the system administrator's personal discretion but more in accordance with the organization's protection guidelines [44]. RBAC is being considered as part of the emerging SQL3 standard for database management systems, based on its implementation in Oracle 7[103]. Many RBAC practical applications have been implemented [103].

However, there is a consistency problem when using RBAC management. For instance, if there are hundreds of permissions and thousands of roles in a system,

it is very difficult to maintain consistency because it may change the authorization level, or imply high-level confidential information to be derived when more than one permission is requested and granted.

The permissions assigned to a role by administrators may conflict. For example, the permission for approving a loan in a bank is conflicts with the permission of funding a loan. These two permissions cannot be assigned to a role; however, because of role hierarchies, a role may still have these permissions even if they have been revoked from the role. In the latter case, a user with this role is able to access objects in the permission and has operations on the objects. Thus, there are evident problems with the processes of assigning and revocation.

Authorization granting problem – How to check whether a permission is in conflict with the permissions of a role?

Authorization revocation problem – How to find whether permissions of a role have been revoked from the role or not?

For example, Figure 7.2 shows a system administrative role (BankSO) in a bank to manage regular roles such as AUDITOR, TELLER, ACCOUNT_REP and MANAGER. Role MANAGER inherits AUDITOR and TELLER. ACCOUNT_REP has a SSD relationship with AUDITOR as well as a DSD relationship with TELLER.

The administrative role BankSO can assign audit permission or cash operation permission to a role but not both, otherwise it compromises the security of a bank system. Our aim is to provide relational algebra algorithms to solve the problems and then automatically check conflicts when assigning and revoking.

Based on the database and its tables such as ROLES, SEN-JUN in the paper[133], this chapter is going to develop formal approaches to check the conflicts and thereby

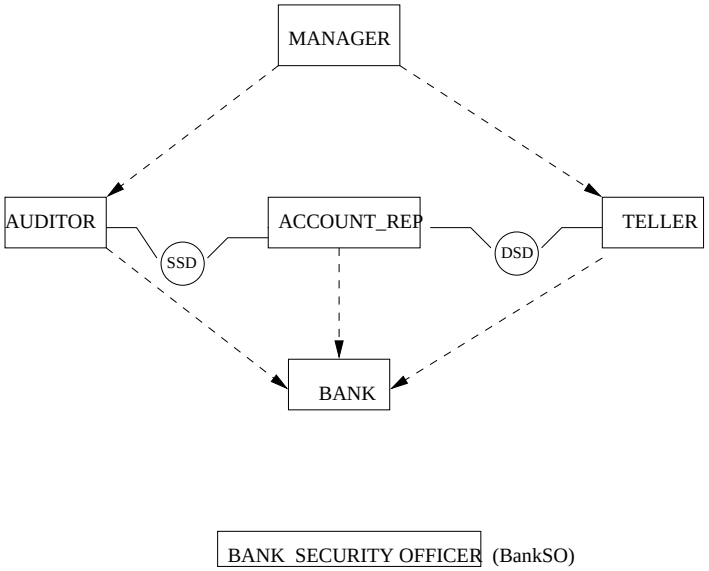

Figure 7.2: Administrative role and role relationships in a bank

help allocate the permissions without compromising the security. The formal approaches are based on relational structure and relational algebra operations. To my knowledge, this is the first attempt in this area to develop formal approaches for permission allocation and conflict detection.

The ROLES relation in Figure 7.2 is in Table 7.1. The attribute TELLERC shows whether the role TELLER conflicts with the RoleName in the relation or not. For instance, in the third tuple, a user with the role TELLER has conflicts with the role AUDITOR.

RoleName	MANAC	AUDC	ACCOUNT_REPC	TELLERC
MANAGER	0	0	0	0
AUDITOR	-1	0	-1	-1
ACCOUNT_REP	-1	-1	0	-1
TELLER	-1	-1	-1	0

Table 7.1: The relation ROLES in Figure 7.2

SEN-JUN - This is a relation of roles in a system. Senior is the senior of the two roles. Table 7.2 expresses the SEN-JUN relationship in Figure 7.2.

Senior	Junior
MANAGER	AUDITOR
MANAGER	TELLER
TELLER	BANK
AUDITOR	BANK

Table 7.2: SEN-JUN table in Figure 7.2

The new tables like PERM and ROLE_PERM are needed.

PERM - This is a relation of {PermName, Oper, Object, ConfPer }:

PermName is the primary key for the table, and is the name of the permission in the system.

Oper is the name of the operation granted. It has information about the object that the operation is granted on.

Object is the database item that can be accessed by the operation. It can be a database, a table, a view, an index or a database package.

ConfPer is a set of permissions that conflicts with the PermName in the relation.

For example, a staff member in a bank cannot have both permissions of approval and funding as well as both permissions of audit and teller. The relation of PERM can be expressed as Table 7.3.

PermName	Oper	Object	ConfPerm
Approval	approve	cash or check	Funding
Funding	invest	cash	Approval
Audit	audit	record	Teller
Teller	transfer	cash	Audit

Table 7.3: An example of the relation PERM

ROLE-PERM - is a relationship between the ROLES and the PERM, listing what permissions are granted to what roles. It has two attributes:

RoleName is a foreign key RoleName from the table ROLES.

PermName is a foreign key PermName from the table PERM which is assigned to the role.

Where the permission Approval is assigned to the role TELLER and the permission Funding to role MANAGER, Table 7.4 expresses the permission-role relationship.

RoleName	PermName
MANAGER	Funding
TELLER	Approval

Table 7.4: An example of ROLE-PERM table

Based on these relations, the Authorization granting algorithm and revocation algorithms are descriped in this chapter.

The chapter is organized as follows. The relational algebra-based authorization granting and revocation algorithms developed in the previous work are recalled. Comparisons to related work are discussed in section 3. The extensions of the algorithms are described in section 4 and the conclusions are in section 5.

7.2 Authorization granting and revocation algorithms for PRA

Granting and revocation algorithms for PRA based on relational algebra are introduced in this section. Details can be found from [133]. The notion of a *Prerequisite conditionp, Can-assignp* and *Can-revokep* mentioned in the chapter is a key part in the processes of permission_role assignments. The *Prerequisite conditionp* is used to test whether or not permission can be assigned to roles while the *Can-assignp* is

used to verify what role range's permissions an administrator can assign.

For a given set of roles R let CR denote all possible prerequisite conditions that can be formed using the roles in R. Not every administrator can assign permission to a role. The relation of **Can-assignp** $\subseteq AR \times CR \times 2^R$ provides what permissions can be assigned by administrators with prerequisite conditions, where AR is a set of administrative roles.

For example, the meaning of *Can-assignp* (x, y, Z) is that a member of the administrative role x can assign a permission whose current membership satisfies the prerequisite condition y to be a member of roles in range Z.

Permission-role assignment (PRA) is authorized by the *Can-assignp* relation. Table 7.5 shows the *Can-assignp* relations with the prerequisite conditions in the example.

Admin.role	Prereq.ConditionP	Role Range
BankSO	BANK $\wedge \overline{TELLER} \wedge \overline{AUDITOR}$	[ACCOUNT_REP, ACCOUNT_REP]
BankSO	BANK $\wedge \overline{TELLER} \wedge \overline{ACCOUNT\_REP}$	[AUDITOR, AUDITOR]
BankSO	BANK $\wedge \overline{AUDITOR} \wedge \overline{ACCOUNT\_REP}$	[TELLER, TELLER]

Table 7.5: Can-assignp relation in Figure 7.2

Supposing an administrator role ADrole wants to assign a permission p_j to a role r with a set of permissions P. P^* is an extension of P, $P^* = \{p | p \in P\} \cup \{p | \exists r' \in R, r' < r, (p, r') \in PA\}$. There are two major steps in the following permission granting algorithm. The first step is to check whether the ADrole can assign the permission p_j to r or not. The set of Prerequisite conditionp associated with ADrole can be obtained from the table *Can-assignp* while the set of roles associated with permission p_j is obtained from the table *ROLE-PERM*. The ADrole can build the

membership of permission p_j and role r only if there is a role in both sets. This means permission p_j satisfies the prerequisite condition. The second step is to determine whether the permission p_j conflicts with the permissions of r or not, or in other words, whether p_j conflicts with permission set P^* or not. The set of conflicting permissions of p_j can be retrieved from table *PERM*. Permission p_j conflicts with role r if the intersection of the set and P^* is not empty.

There are related subtleties that arise in RBAC concerning the interaction between granting and revocation of permission-role membership. A relation **Can-revokep**$\subseteq AR \times 2^R$ provides which permissions in what role range can be revoked. Table 7.6 gives an example of the *Can-revokep* relation. We have two revocation algorithms, one is a weak revocation algorithm that is for explicit members of a role only, while the other one is a strong revocation algorithm that is used to delete explicit memberships between permissions and roles as well as implicit memberships.

Admin.role	Role Range
BankSO	[Bank, MANAGER)

Table 7.6: An example of Can-revokep

The meaning of *Can-revokep* $(BankSO, [Bank, MANAGER))$ in Table 7.6 is that a member of the administrative role BankSO can revoke the membership of a permission from any role in [Bank, MANAGER).

Two algorithms were developed for revocation of a permission p_j from a role r by an administrative role ADrole [133]. In the weak revocation approach, only whether or not permission p_j is an explicit member of role r needs to be determined. Operation of the weak revocation has no effect when the permission p_j is not an explicit member of the role r. The role set associated with p_j is gained from the

relation of *ROLE-PERM* while the role set of role range with ADrole is obtained from the relation *Can-revokep*. The permission p_j can be revoked if the intersection of these two role sets is not empty.

A role still owns a permission of a system, which has been weakly revoked, if the role is senior to another role associated with the permission. To solve the authorization revocation problem, we need strong revocation, which requires revocation of both explicit and implicit membership. Strong revocation of a permission's membership in role r requires that the permission be removed not only from explicit membership in r, but also from explicit and implicit membership in all roles junior to r. Strong revocation therefore has a cascading effect up-wards in the role hierarchy. The first step in the strong revocation algorithm is to test whether p_j is in P^* or not. If the test is negative, that means p_j is neither an explicit member nor an implicit member of the role r. When this case occurs, the strong revocation has no effect for the role. Otherwise, p_j is either an explicit member or an implicit member of r. In this step, the membership of p_j is revoked from r if $p_j \in P$; then the role set of roles that are junior to role r can be retrieved from the relation *SEN-JUN*. For all roles in both the set and P, the relationships between these roles and permission p_j are revoked.

7.3 Related work

There are several related works on relational databases [94, 104].

The interaction between RBAC and relational databases is presented in [94]. Two experiments are described. One is a role-based front end to a relational database with discretionary access control. The other is a role graph to show the roles in a standard relational database. Some relational concepts like roles, users and permis-

sions are provided. This study's model also supports such concepts even though it has a large variety. However, the main difference between this study's algorithms and the scheme in [94] is that this stydy focuses on the solutions of the conflicts of roles and permissions, and the latter focuses on the correlation of RBAC with discretionary access controls. Their work discusses the relationship between roles and discretionary access controls, and they do not address the allocation of permissions to roles without conflicts. In this work, we developed detailed algorithms for allocating roles and permissions and checking their conflicts.

An oracle implementation for permission-role assignment has been proposed in [104]. In [104], the difference between permission-role assignment and the Oracle database management system was analyzed. Furthermore, through prerequisite conditions, the paper demonstrated how to use Oracle stored procedures for implementation. However, the work in this chapter substantially differs from that proposal. Differences are due to the consistency problem that arises in [104]:

It is very difficult to keep the consistency by reflecting security requirements between global network objects and local network objects if there are hundreds of roles and thousands of users in a system.

This problem is completely overcome in our algorithms because the algorithms focus on the conflicts between roles and permissions. The authorization granting algorithms are used to find conflicts and prevent secret information from being derived while the strong revocation algorithms are used to check whether a role still has the permissions of another role.

7.4 Extensions of the algorithms with mobility of permissions

Similar to the mobility of the user-role relationship, permissions can also be assigned to roles as mobile and immobile members [136]. There are four kinds of permission-role membership for a given role x [105].

1: *Explicit Mobile Member EMPx*

$EMPx = \{p, (p, Mx) \in PA\}$

2: *Explicit Immobile Member EIMPx*

$EIMPx = \{p, (p, IMx) \in PA\}$

3: *Implicit Mobile Member ImMPx*

$ImMPx = \{p, \exists x' < x, (p, Mx') \in PA\}$

4: *Implicit Immobile Member ImIMPx*

$ImIMPx = \{p, \exists x' < x, (p, IMx') \in PA\}$

A **prerequisite conditionPM** is evaluated for a permission p by interpreting role x to be true if

$$p \in EMx \vee (p \in ImMx \wedge p \notin EIMx)$$

and $\bar{x}$ to be true if

$$p \notin EMx \wedge p \notin EIMx \wedge p \notin ImMx \wedge p \notin ImIMx$$

In other words x denotes mobile membership (explicit or implicit) and $\bar{x}$ denotes absence of any kind of membership.

For a given set of roles R let CR denote all possible prerequisite conditions with

Admin.role	Prereq.ConditionPM	Role Range
BankSO	BANK	[BANK, BANK]
BankSO	BANK $\wedge\overline{TELLER}$	[AUDITOR, AUDITOR]
BankSO	BANK $\wedge\overline{AUDITOR}$	[TELLER, TELLER]
BankSO	TELLER $\wedge$ AUDITOR	[MANAGER, MANAGER]

Table 7.7: Can-assignp-M in the example

Admin.role	Prereq.ConditionPM	Role Range
BankSO	BANK	[BANK, BANK]
BankSO	BANK $\wedge\overline{TELLER}$	[AUDITOR, AUDITOR]
BankSO	BANK $\wedge\overline{AUDITOR}$	[TELLER, TELLER]

Table 7.8: Can-assignp-IM in the example

mobility of the permission-role relationship that can be formed using the roles in R. Not every administrator can assign a role to a user. The following relations provide what permissions an administrator can assign to mobile members or immobile members with prerequisite conditions.

Can-assignp-M is a relation of $\subseteq AR \times CR \times 2^R$, which is used for permission-role assignments with mobile members; where AR is a set of administrative roles. Permission-role assignments with immobile members are authorized by the relation **Can-assignp-IM** $\subseteq AR \times CR \times 2^R$

Permission-role assignment(PRA) is authorized by *Can-assignp-M* and *Can-assignp-IM* relations. Table 7.7 and Table 7.8 show the *Can-assignp-M* and *Can-assignp-IM* relations with the prerequisite conditions in the bank example.

The meaning of *Can-assignp-M* $(BankSO, BANK, \{TELLER, AUDITOR\})$ is that a member of the administrative role BankSO can assign a permission whose current membership satisfies the prerequisite condition BANK to be a mobile member of roles TELLER and AUDITOR.

Supposing an administrator role ADrole wants to assign a permission p_j to role r with a set of permissions P which has mobile and immobile memberships with r. The p_j has mobile or immobile membership with r if ADrole can assign without conflicts. The following algorithm applies to both mobile and immobile members. P^* is an extension of P, $P^* = \{p|p \in P\} \cup \{p|\exists r', r' < r, (p, r') \in PA\}$.

Authorization granting algorithm

GrantMP(ADrole, P, p_j)

Input: ADrole, role r and a permission p_j.

Output: true if ADrole can assign the permission p_j to r with no conflicts; false otherwise.

Begin:

Step 1. /* *Whether the ADrole can assign the permission* p_j *to* r *as mobile or immobile member or not* */

Suppose $S_{M1} = S_M \cap R$ and $S_{IM1} = S_{IM} \cap R$ where

$$S_M = \pi_{Prereq.ConditionPM}(\sigma_{admin.role=ADrole}(Can - assignp - M))$$

$$S_{IM} = \pi_{Prereq.ConditionPM}(\sigma_{admin.role=ADrole}(Can - assignp - IM))$$

$$R = \pi_{RoleName}(\sigma_{PermName=p_j}(ROLE - PERM))$$

if p_j is a mobile member of r and $S_{M1} \neq \phi$,

then there exists a role $r_1 \in S_{M1}$, such that

$r_1 \in \pi_{RoleRange}(\sigma_{\{ADrole,r\}}(Can - assign - M))$, and $(p_j, r_1 \in PA)$,

/* p_j *is in the range to be assigned as a mobile member by ADrole in Can-assignp-M* */

if p_j is an immobile member of r and $S_{IM1} \neq \phi$,

then there exists a role $r_i \in S_{IM1}$, such that

$r_i \in \pi_{RoleRange}(\sigma_{\{ADrole,r\}}(Can-assignp-IM))$ and $(p_j, r_i \in PA)$

go to step 2 /* *p_j is in the range to be assigned as an immobile member by ADrole in Can-assign-IM* */

else

RETURN false and stop. /**the admini.role has no right to assign the role r_j as a mobile or immobile member to R* */

Step 2. /**whether the permission p_j conflicts with permissions of r or not**/

Let

$$ConfPermS = \pi_{ConfPerm}(\sigma_{PermName=p_j}(PERM))$$

/* *It is the conflicting permission set of the permission p_j* */

if $ConfPermS \cap P^* \neq \phi$,

then

RETURN false; /* *p_j is a conflicting permission with role r* */

else

RETURN true. /* *p_j is not a conflicting permission with r* */

This algorithm provides a way to decide whether a permission can be assigned to a role as mobile or immobile member. For a mobile member, S_{M1} cannot be empty, and for an immobile member, S_{IM1} cannot be empty.

Now we consider revocation of permission-role membership. Similar to the *Can-assignp-M* and *Can-assignp-IM* relations in granting a permission to a role, there are *Can-revokep-M* and *Can-revokep-IM* relations.

Relations **Can-revokep-M** $\subseteq AR \times CR \times 2^R$ and **Can-revokep-IM** $\subseteq AR \times CR \times 2^R$ show which role range of mobile membership and immobile membership administrative roles can revoke respectively, where AR is a set of administrative roles. $\diamond$

The meaning of *Can-revokep-M*$(ShopSO, SHOP, [SHOP, MANAGER))$ in Table 7.9 is that a member of the administrative role ShopSO can revoke mobile membership of a permission from any role in the [SHOP, MANAGER) subject to the prerequisite condition SHOP. *Can-revokep-IM* is similar with respect to immobile membership.

Admin.role	Prereq.ConditionPRM	Role Range
ShopSO	SHOP	[SHOP, MANAGER)

Table 7.9: Can-revokep-M

Admin.role	Prereq.ConditionPRM	Role Range
ShopSO	SHOP	[SHOP, MANAGER)

Table 7.10: Can-revokep-IM

The evaluation of a prerequisite condition for the revoke model is different from the grant model. In the revoke model a **prerequisite conditionPRM** is evaluated for a permission p by interpreting role x to be true if

$$p \in EMx \vee p \in EIMx \vee p \in ImMx \vee p \in ImIMx$$

and $\bar{x}$ to be true if

$$p \notin EMx \wedge p \notin EIMx \wedge p \notin ImMx \wedge p \notin ImIMx$$

Due to role hierarchy, a role x' has all permissions of role x when $x' > x$. A user with two roles $\{x', x\}$ still has the permissions of x if only to revoke x from the user. To solve the authorization revocation problem along with mobility of permission, we need to revoke the explicit member of a permission first if a role is an explicit member, then revoke the implicit member.

Following are two algorithms for revocation of a permission p_j as mobile or immobile members from a set of permissions P by an administrative role ADrole, where P is a set of permissions which are assigned to a role r. The first one is the weak revocation algorithm and the second is the strong revocation algorithm. The weak revocation only revokes explicit mobile and immobile memberships from r and does not revoke implicit mobile and immobile memberships, but the strong revocation revokes both explicit and implicit mobile and immobile members.

Weak revocation Algorithm

Weak_revokeMP(ADrole, r, p_j)

Input: ADrole, a roles r and a permission p_j.

Output: true if ADrole can weakly revoke role p_j from r; false otherwise.

Begin:

if $p_j \notin P = \{p|(p, r) \in PA\}$,

RETURN false; /* *there is no effect with the operation of the weak revocation since the permission* p_j *is not an explicit member of the role* r*/

else /* p_j *is an explicit member of* r */

Case1: p_j is an mobile member of r,

$$Roleswithp_j = \pi_{RoleName}(\sigma_{PermName=p_j}(ROLE - PERM))$$

/* *Roles with permission* p_j */

$$PreM = \pi_{Prereq.ConditionPRM}(\sigma_{admin.role=ADrole}(Can - revokep - M))$$

/**Prerequisite condition with ADRole* */

if $RP = Roleswithp_j \cap PreM \neq \phi$

$$RevokeRangeM = \pi_{RoleRange}(\sigma_{admin.role=ADrole}(Can - revokep - M))$$

if $RR = Roleswithp_j \cap RevokeRangeM \neq \phi$,

RETURN, true. /* *the mobile member* p_j *is revoked* */

else RETURN false;

/* *the mobile member* p_j *cannot be revoked since the role* r *is not in the role range to be revoked* */

else RETURN false and stop.

/**The* p_j *does not satisfy the prerequisite conditions** /

Case 2: **if** p_j is an immobile member of r

$$PreIM = \pi_{Prereq.ConditionPRM}(\sigma_{admin.role=ADrole}(Can - revokep - IM))$$

/**Prerequisite condition with ADRole** /

if $RPI = Roleswithp_j \cap PreIM \neq \phi$

$$RevokeRangeIM = \pi_{RoleRange}(\sigma_{admin.role=ADrole}(Can - revokep - IM))$$

if $RRI = Roleswithp_j \cap RevokeRangeIM \neq \phi$,

RETURN true, /* *the immobile member* p_j *is revoked* */

else RETURN false ; /* *the immobile member* p_j *cannot be revoked* */

else RETURN false and stop.

/* *The p_j does not satisfy the prerequisite conditions* */

The weak revocation algorithm can be used to check whether an administrator can weakly revoke mobile and immobile memberships from roles or not. The following result is gained with the weak revocation algorithm.

A role still owns a permission of a system, which has been weakly revoked, if the role is senior to another role associated with the permission. To solve the authorization revocation problem, we need strong revocation, which requires revocation of both explicit-implicit membership and mobile-immobile memberships. Strong revocation of a permission's membership in role r requires that the permission be removed not only from explicit mobile and immobile membership in r, but also from explicit, and implicit mobile and immobile membership in all roles junior to r.

Strong revocation algorithm

Strong_revoke(ADrole, r, p_j)

Input: ADrole, a role r and a permission p_j.

Output: true, if it can strongly revoke the permission p_j from r; false otherwise.

Begin:

if $p_j \notin P^*$,

RETURN false;

/* *there is no effect of the strong revocation since the permission is not an explicit and implicit member of the role r* */

else,

1. **if** $p_j \in P$, **do** Weak_revoke(ADrole, r, p_j);

/*p_j *is weakly revoked from* r *as mobile or immobile* */;

2. Suppose

$Jun = \pi_{Junior}(\sigma_{Senior=r}(SEN - JUN))$,

for all $y \in Jun \cap P$,

if $p_j \in R$ is a mobile member of the role y, **do** Weak_revoke(ADrole, y, p_j) as $p_j \in EMy$;

if $p_j \in R$ is a immobile member of the role y, **do** Weak_revoke(ADrole, y, p_j) as $p_j \in EIMy$;

/**the permission* p_j *is weakly revoked from all such* $y \in Jun \cap P$*/.

if all the weak revocations are successful,

RETURN true;

otherwise,

RETURN false. /* if one weak revocation cannot finish*/

7.5 Conclusions

This chapter has provided new authorization allocation algorithms for mobility of permission-role assignments that are based on relational algebra operations. They include the authorization granting algorithm, weak revocation algorithm, and strong revocation algorithm. The algorithms can automatically check conflicts when granting more than one permission to a role in a system. They can prevent users associated with roles from accessing unauthorized use of facilities when the permissions

of the roles are changed within the organization and demand the modification of security rights. The permissions can be allocated without compromising the security in RBAC and provide secure management for systems. Finally, the related work in this area has been discussed.

Chapter 8

Protecting disseminative information in E-learning

This chapter aims to substantially provide a foundation for developing appropriate security solutions for organizations' secure dissemination of digital information in *XML* documents. An application-level secure architecture is developed based on references to both server-side and client-side. The architecture provides control and tracking capabilities for dissemination and usage of digital information while others provide only a tracking capability. The outcomes of this chapter can immediately apply to prevent unauthorized dissemination of digital content in *XML* documents amongst agents of universities regardless of their intention or possession of the digital information and will contribute to higher security in E-learning.

The information in this chapter is based on a published paper [116].

8.1 Introduction

E-learning organisers prefer to disseminate their messages as widely as possible, and at the same time authorised people can only access some disseminative information. It is often the case that the information contained in *XML* documents with different

sensitivity degrees must be selectively shared in E-learning environments. It becomes a challenge to protect the *XML* documents from access such as reading and updating which has been disseminated out from the organisations.

With recent advances in the Internet and information technologies, we find ourselves accustomed to the pervasive and convenient availability of digital information. The proliferation of inexpensive digital equipment and the Internet has expedited this availability to a scale imagined. E-learning, as a product of current advanced technology, has brought great benefits to both education organisers and students since students can access digital information provided by E-learning organisers without space and time limits [32, 61]. E-learning strategies are required to satisfy the changes in a competitive and commercial industry market [110, 83]. The new strategies will reshape the role of education and create enduring advantages for both students and universities. Digital information sent by the organisers to students or agents may not be further disseminated for commercial reasons. Therefore unauthorized dissemination of digital content has emerged as one of the most problematic and challenging issues in information security within E-learning.

Digital information in E-learning *XML* documents is divided into two types based on their purposes: Payment-Based Type (PBT) and Payment-Free Type (PFT). An example of PBT is course information, where students can access the study materials if they have paid the tuition fee; and the information of learning structures and major components are examples of PFT. In PBT, a payment is required in order to access digital information and security breaches of digital assets result directly in financial loss. In PFT, dissemination of digital information does not require payment, but as the PFT example mentioned, organisers will not be happy to show competitors the learning plans and components which must be controlled to satisfy confidentiality or

other security requirements. Unlike the commercial distribution environment, there are situations in which a payment function is not required and higher distribution security is the primary concern. In the intelligence community, for instance, digital information is often disseminated to organizations via various agents. For instance, the University of Southern Queensland (USQ) may wish to distribute a document in digital form to its education agents in such a manner that the received digital information is not revealed either intentionally or accidentally, to other malicious organizations. Similar situations can exist in the commercial sector. In recent business-to-business (B2B) e-commerce, it is common for a computer organization to distribute information digitally to its several partners. The challenge is to prevent further distribution of the digital information by the partners to others. For instance, IBM could disseminate technical descriptions in digital form to different suppliers who provide the specific parts of IBM computers. However, IBM would not like to disseminate the digital information amongst suppliers regardless of their intention or possession of the digital information. Digital content providers have put much effort into protecting digital information from unauthorized distribution [134]. However, no systematic study has been done for controlling digital information dissemination [23, 36].

The most successful E-learning models in the future will be hybrid E-learning networks that are combinations of academic, professional and corporate content [109]. Human interaction administration, as a shortcoming of E-learning, is a critical component of the E-learning market, especially in several organisations' collaborative environments. There are situations in which E-learning strategies and plans are disclosed to other competitors due to insufficient protection management systems. E-learning has not been able to securely control the dissemination of strategies and

plans. It is still an open question how efficiently technical skills can be trained to protecting the disseminative information in distributed E-learning environments [113, 109].

The main objective of this chapter is to develop a secure system for digital information dissemination by defining the security architectures that can provide control ability to protect the disseminated digital information. The security architectures should support tracking ability on the disseminated digital information. More specifically, proposed security architectures should make it difficult or useless for recipients to re-disseminate the received digital information if not authorized, regardless of their intention. In addition, they should make it difficult for new recipients to access the illegitimately re-disseminated digital information that they possess.

This chapter presents authorization models which adopt usage control to manage access to digital information and secure architectures to protect against malicious dissemination. In today highly dynamic, distributed environment, obligations and conditions of new hosts are decision factors for the management of digital documents. Because of the complex environment of the Internet, users are required to obey obligations and satisfy conditions and ongoing control with different security policies. In the usage control models, Authorizations, Obligations and Conditions are used to build a secure architecture. The ongoing control provides dynamic access verification for digital documents.

This chapter is organized as follows. The next section presents the motivation of the chapter including the background of disseminative digital information. Section 3 shows a proposed authorization models for usage control. Section 4 discusses

how to build secure architectures for digital information by using reference monitors in detail. Section 5 compares this work with previous work on digital document security. The difference between this work from others is presented. Finally section 6 concludes.

8.2 Motivation

Many documents including study books and course specifications are in digital form in E-learning. These documents are usually disseminated to other organisations such as education agents and collaborative offices due to the efficient response, and they should be securely protected from accidental or malicious attacks, even if the digital information has been delivered to other organisations. The digital information of PFT is different from that of PBT. In the PBT situation, digital information leakage is acceptable and desired [23] while this may be rejected for a PFT message. The objective of a PBT message is to distribute as many copies as possible to attract payment for each copy. On the other hand, the dissemination of PFT information should be limited. The number of copies of PBT is larger than the copies of PFT. Therefore, the solutions for information dissemination of PFT differ from the solutions of PBT.

Recently, dissemination of digital information has obtained increasing attention from both commercial and non-commercial perspectives. The disseminative digital information should be managed by a distributor who can limit recipients' access to the information. The recipients of the digital information need an access permission to the information, and some addition restrictions are need such as the recipients not being able to modify and copy the information. There may be some undesirable leakage of digital information in E-learning, and a secure protocol is required to con-

trol the recipients access to a digital message but also to trace the further behaviour around the message such as who has re-disseminated and what has occurred. For instance, the distance education office (*DEO*) at USQ sent out a new study plan to its collaborative organisers at HK with a high confidentiality level. The *DEO* does not like the organisers to resend the new plan nor any changes. Hence, several protection requirements are necessary:

1. Access control management. There are many access approaches in traditional access control, and we should chose an efficient one for the E-learning since most of students are not willing to buy high cost equipments.

2. Trace what happened to the digital information. An application level secure architecture is needed to track the information which should not create too much work for either server side or client side.

3. Revocation schemes are an important feature of E-learning systems. They take away the access permissions. There are different revoking schemes; among them are strong and weak revocations, cascading and noncascading revocations, as well as grant-dependent and grant-independent revocations [133].

4. Constraints are an important factor in E-learning for laying out higher-level organizational policies [131]. They define whether or not the access permission or revocation process is valid.

8.3 Authorization models

Next, authorization models for digital documents adopting usage control are discussed in this section. Based on three decision factors: authorizations, obligations,

and conditions, a family of core models for usage control is developed. Core models focus on the enforcement process and do not include administrative issues. We assume there exists a usage request for a digital information object. Decision-making can be done either before (pre) or during (ongoing) the exercise of the requested right. Decision-making after the usage has no influence on the decision of the current usage. Based on these criteria, there are six possible case spaces as a core model for usage control: pre-Authorizations, ongoing-Authorizations, pre-Obligations, ongoing-Obligations, pre-Conditions and ongoing-Conditions. The details concepts of these models have been explained in Chapter 4.

8.4 Security architecture

In this section, architecture solutions for digital access control based on reference monitors are discussed. Reference monitors have been discussed extensively in the access control community. Subjects can access digital objects only through the reference monitor since it provides control mechanisms on access to digital documents.

8.4.1 Structure of reference monitor

ISO has published a standard for an access control framework by using reference monitors [67]. Based on the standard, dissemination reference monitors consist of Usage Decision Facility (UDF) and Usage Enforcement Facility (UEF) as shown in Figure 8.1. Each facility includes several functional modules.

UEF includes *Customization, Monitor and Update modules* and UDF includes *authorization, conditions and obligations decision modules.* When a subject sends an access request through the *Customization module* to the *Authorization module*,

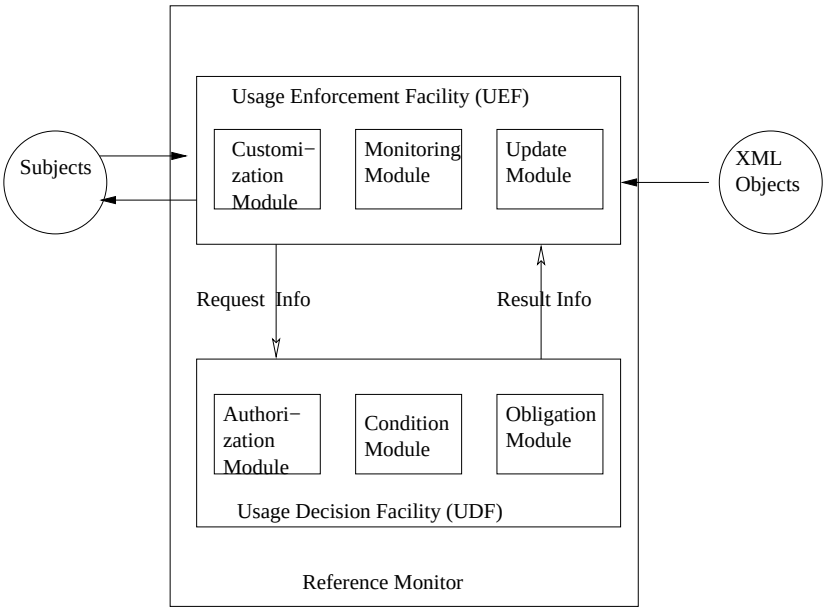

Figure 8.1: XML reference monitor

the *Authorization module* verifies the authorization process and checks whether the request is allowed or not. It may return yes or no or metadata information of the authorization result. This metadata information can be used for approved access to digital objects by the *Customization module* in UEF. The *Condition module* is used to make a decision for whether the conditional requirements are satisfied or not. The *Obligation module* is applied to verify whether obligations have been performed or not before or during the requested usage. When any obligation is changed, it must be monitored by the *monitor module* and the result has to be resolved by the *Update module* in UEF. Applications of these modules rely on object systems requirements.

8.4.2 Architectures

There are two kinds of reference monitors: Server-side Reference Monitor (SRM), and Client-side Reference Monitor (CRM). Servers provide digital documents and clients require access to digital messages. Like a traditional reference monitor, an SRM works in a server system environment and manages access to digital informa-

tion in the server. On the other hand, a CRM works in the client environment and controls access to digital messages when it works as a server for other clients. For example, the client acts as a server when the digital document is disseminated to other users. SRM and CRM can coexist within a system. For real implementations, both CRM and SRM should be used for better security. Architectures are here analysed according to reference monitors on the server side only (SRM-only), on the client side only (CRM-only) and on both server and client sides (SRM & CRM).

SRM-Only Architecture

A system with SRM-only facilitates work on the server side only to control subjects' access to digital objects. In this case digital objects may or may not be stored on the client-side. If the digital object is allowed to reside on the client-side, it means the saved client copy of the digital object is no longer valid and does not have to be controlled. It can be used and changed freely at the client-side. For example, a student payment bank statement can be saved at a client's local machine for his/her records and the server (bank) does not care how the copy will be used by the client since the bank keeps the original account information safe. However if the digital document or some parts of the document have to be protected and controlled centrally, the digital objects must remain in server-side storage and are not allowed to be stored on the client-side. This is the main topic of traditional access control and trust management systems.

CRM-Only Architecture

No reference monitor exists on the server-side in a system with CRM-only environment. Rather, a reference monitor exists at the client system for controlling usage of disseminated digital documents. In this environment digital objects can be

stored either centrally or locally. The usage of digital objects saved at the client-side on behalf of a server is still under the control of CRM. Distributed digital documents are associated with certain usage rules and users may need to prove they have sufficient credentials to access the document.

SRM & CRM Architecture

With both SRM and CRM, this architecture can provide a comprehensive access control. SRM may be used for distribution related control while CRM can be used for digital document dissemination. For instance, in SRM, digital objects can be pre-customized for distribution. The pre-customized digital objects can be further controlled and customized by CRM. As a result, a server can restrict or eliminate unnecessary exposure of digital objects that do not have to be distributed. If a user requests certain digital documents that includes some secret information, SRM can pre-customize the requested objects before distribution such that the distributed version of the objects does not include any secret information. If the document cannot be disseminated, the CRM at client side can do this work.

Thus, the SRM and CRM architecture provides a solution for restricting access to digital documents and protecting digital documents from malicious dissemination.

8.5 Comparisons

Related work has been done on secure and selective dissemination of *XML* documents [12] and on securing *XML* Web services [37].

Elisa and Elena [12] proposed an access control system supporting selective distribution of *XML* documents among possible large user communities by using a

range of key distribution methods. An approach based on cryptograph is designed to allow sending the same document to all users, and to enforce stated access control policies. This proposal is different from this study's schemes. It focused on key distribution methods to protect *XML* documents. Therefore, it only discussed the management on the server side and without any mention of how to control the *XML* document when users get keys. By contrast, this work provides options that can deal with digital documents on both server and user sides.

Securing *XML* Web services was described by Damiani, Vimercati and Samarati in 2002 [37]. Two experiments are discussed. One is restricting access to an *XML* Web service to authorized users. Another one is protecting the integrity and confidentiality of *XML* messages exchanged in a Web service environment. The authors introduce SOAP highlights, and how to use SOAP headers for credential transfer and access control. The main difference between this study's scheme and the work in [37] is that we focus on a systematic level for digital documents by using a usage control model and by considering a solution for different kinds of authorizations, whereas the latter is a discussion of providing a secure infrastructure for *XML* Web services.

8.6 Conclusions

This chapter has analysed the dissemination of payment-free-type digital information in E-learning, and discussed access models and architectures by using usage control. Different kinds of models are built for digital documents have been analysed. To protect digital documents from malicious dissemination, we have analysed reference monitors on both server and client sides and obtained several secure architecture solutions. The work in this chapter has significantly extended previous work

in several aspects. The proposed methods can be used to control digital documents in the E-learning environment since they provide robust access control for digital information and they can protect sensitive messages from dissemination.

Chapter 9

Conclusions and future work

This chapter lists the contributions of this dissertation and introduces future work. The summary of this dissertation is presented in section 9.1 and section 9.2 explains and suggests future research.

9.1 Summary

In today's context, many important factors motivate moving access control from servers to client devices. Software technology, including access schemes and access control management are an important dimension of e-learning. However, by compiling the access control policies into the data encryption, existing client-based access control solutions minimize the trust required on part of the client for the price of a rather static way of sharing data.

Unlike existing traditional access control models, we presented the usage access control models on *XML* documents, *XML* databases and e-learning applications, this clear distinction allows for the provision of a dynamic access control in these areas. We built a connection between *XML* documents and usage access control, developed authorization models with usage access control for the access management of *XML*

documents. We also provided a fundamental analysis for a usage control model on *XML* databases and new authorization algorithms for permission-role assignments.

In Chapter 2, we introduced the different technologies which can be used to build our solution. First, we presented *XML* and its two main validation specification mechanisms Document Type Definition (*DTD*) and *XML* Schema. More precisely, we described an *XML* document and the correspondence with its *DTD* and *XML* Schema while offering the features of *DTD* and *XML* Schema. Second, we provided an overview of different access control and access control system for *XML*. Finally, We presented usage access control models and the properties of decision continuity for usage access control, which can protect information distributed on the web.

In Chapter 3, we described *XML* and web services security standard. Most specifications for these standards are introduced. Several specifications progressed toward providing a comprehensive standards framework for securing *XML*-bases applications have been presented. Each standard that connects with protecting *XML* based documents has been discussed. We also briefly describe the relations with these standards based on existing technologies.

We proposed in Chapter 4 to adopt usage control authorization models to manage access both at the instance-level and at schema-level *XML* documents. First, we explained the different *XML* access control models which have been proposed and their limitations, such as encryption and decryption skills, and traditional access control models. In order to alleviate these limitations, we presented usage access control models for *XML* documents. We took advantages of usage access control for continuity of access decisions and mutability of subject attributes and object attributes. Moreover, we proposed authorization models for the *DTD* and *XML* documents adopting usage control. Based on the involvement of three decision fac-

tors: authorizations, obligations, and conditions, we developed pre-Authorizations, ongoing-Authorizations, pre-Obligations, ongoing-Obligations, pre-Conditions and ongoing-Conditions six models.

In Chapter 5, we presented a usage control model to protect information distributed on the web, which allows the access restrictions directly at Schema-level and *XML* document-level. First, we addressed two *XML* validations: *XML DTD* and *XML* Schema. We analysed and compared these two validations. *XML* Schema was shown to be more effective than *XML DTD*. Then we introduced two authorization models built for *XML* Schemas and *XML* documents. Finally, we designed an algorithm based on these models.

In Chapter 4 and Chapter 5, we only discussed about usage access models to access *XML* documents. In Chapter 6 we provided usage control to access *XML* databases. This is believed to be the first approach dealing with usage access control in *XML* databases. First, we explained the details of *XML* and *XML* databases. *XML* databases are used to be stored in the *OODB* system. Therefore, we addressed *OODB* authorization models. Then comparing with the *OODB* authorization model and usage control models for accessing *XML* databases. Finally, we provided some functions and developed *XML* databases with usage control models.

In Chapter 7, we analysed the problems during the procedures of permission-role assignments in distributed collaborative environments and provided new authorization allocation algorithms for mobility of permission-role assignments. The algorithms were extended to the case of *PRA* with the mobility of a permission-role relationship.

In Chapter 8, we provided a foundation for developing appropriate security solu-

tions for organizations' secure dissemination of digital information. An application-level secure architecture has been developed based on the references on both the sever-side and client-side. We designed the architecture which can control and track capabilities for dissemination and usage of digital information while others provided only tracking capability. This can apply to prevent unauthorized dissemination of digital content amongst agents of universities regardless of their intention or possession of the digital information in E-Learning.

9.2 Future work

Based on the research in this dissertation, we propose the following future research directions and issues.

9.2.1 Improvement of *XML* databases with usage access control models application

We have given a technique of specifying a complex authorization model for *XML* data using usage access control and a model for authorization for complex data structures previously developed for object oriented databases. However, for a complex *XML* Schema and a large set of *XML* documents with many elements, will need to give many privileges. It might therefore be more efficient for accessing the *XML* databases system. In the future, we will investigate the automatic generation of the *AOS* from an *XML* Schema using *XSLT* and other *XML* tools.

9.2.2 Extension of authorization approaches for usage access control

In this dissertation we presented formal authorization approaches for *XML* documents and *XML* databases with usage access control models. But safety analysis is still an undecidable problem in usage access control models. For pre-authorization core models, every access can potentially enable additional permissions due to the mutability of attributes in usage access control. For condition core models of usage access control, as system state changes caused by environmental information are not captured in usage access control core models, safety is a function of the system environment. For obligation core models, an obligation of an access is an action that can be related to the subject requesting the access, or to some other subjects, and therefore, a usage policy may include more than two parameters. In the future, we would like to extend authorization approaches with safety analysis in the core usage access control models. We must also determine how safety can be used to enforce a usage access control management system.

9.2.3 E-Learning application with usage access control

There is an architecture of usage access control for disseminative information in E-Learning system. This provided a way to use usage access control to manage disseminative information in E-Learning. However, the use of usage access control for E-Learning systems is still far from being achieved. We like to develop algorithms based on the models and architectures and applications of the algorithms in real implementation. Then we will extend the results to generic E-Learning systems.

9.2.4 Implementation issues

We would like to build tools that can check the syntax and the semantics of authorization approaches for usage access control. These tools might provide visual support for authorization approaches to make them more efficient. The visual tool of formal authorization approaches, for example, would display all of the components such as elements and attributes that can be used in databases. In addition, this shows what components are used and are available for the authorization approaches. Furthermore, using the visualization tool, system administrators can easily check the current status of components in systems.

Bibliography

[1] International workshop on practice and theory in public key cryptography (pkc '98). 1346, 1998, Yokohama, Japan.

[2] XML Key Management Specification 2.0(XKMS). 10 March 2002, http://www.w3.org/TR/xkms2/.

[3] XML-Signature Syntax and Processing. February 2002, http://www.w3.org/TR/xmldsig-core/.

[4] XML Encryption Syntax and Processing. March 2002, http://www.w3.org/TR/xmlenc-core/.

[5] Web Services Security: SOAP Message Security 1.0 (WS-Security 2004). March 2004, http://docs.oasis-open.org/wss.

[6] eXtensible Access Control Markup Language (XACML). 1 Feb 2005, http://docs.oasis-open.org/xacml/2.0/.

[7] Security Assertion Markup Language (SAML) 2.0 Technical Overview. 20 Feburary 2005, http://www.oasis-open.org/committees/security.

[8] Public-Key Infrastructure(X.509). http://www.ietf.org/html.charters/pkix-charter.html.

[9] Ahn G. and Sandhu R. Role-based authorization constraints specification. *Information and System Security*, 3(4):207–226, 2000, New York, NY, USA.

[10] Arenas M. and Libkin L. A normal form for xml documents. *ACM Trans. Database Syst.*, 29(1):195–232, 2004.

[11] Baskerville R. Hacker wars: E-collaboration by vandals and warriors. *International Journal of e-Collaboration*, 2(1):1–16, 2006.

[12] Bertino E. and Ferrari E. Secure and selective dissemination of xml documents. *ACM Trans. Inf. Syst. Secur.*, 5(3):290–331, 2002.

[13] Bertino E. and Sandhu R. Database security - concepts, approaches, and challenges. *IEEE Transactions on dependable and secure computing*, 2(1), January-March, 2005.

[14] Bertino E., Castano S. and Ferrari E. Securing xml documents: the author-x project demonstration. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, page 605. ACM Press, 2001.

[15] Bertino E., Castano S. and Ferrari E. Securing xml documents with author-x. *IEEE Trans, Knowledge and Data Eng*, pages 4–23, 2004.

[16] Bertino E., Castano S., Ferrari E. and Mesiti M. Specifying and enforcing access control policies for xml document sources. *World Wide Web Journal*, 3(3):139–151, 2000.

[17] Bertino E. P., Samarati P. and Jajodia S. An extended authorization model for relational databases. *IEEE Transactions On Knowledge and Data Engineering*, 9(1):145–167, 1997.

[18] Bertino E.,Castano S., Ferrari E. and Mesiti M. Controlled access and dissemination of xml documents. In *Proceedings of the second international workshop on Web information and data management*, pages 22–27. ACM Press, 1999.

[19] Bertion E. Protecting xml documents. In *Proceedings of Computer Software and Applications Conference*, pages 132–133. Taiwan, 2000.

[20] Boris K. and Jajodia S. Concurrency control in multilevel-secure databases based on replicated architecture. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, pages 153–162. ACM Press, 1990.

[21] Bourret R. Xml database products. 7 August, 2003.

[22] Box D. *Simple Object Access Protocol (SOAP) 1.1.* World Wide Web Consortium (W3C), Cambridge, MA, USA, 2000, http://www.w3.org/TR/soap.

[23] Brad Cox. Superdistribution: Objects as property on the electronic frontier. 1996.

[24] Bray T., Paoli J., Sperberg M. and Maler E. *Extensible Markup Language (XML) 1.1 (Second Edition).* World Wide Web Consortium (W3C), Cambridge, MA, USA, 2000, http://www.w3.org/TR/REC-xml.

[25] Brofferio S.C. A university distance lesson system: experiments, services, and future developments. *IEEE Transactions on Education*, 41(1):17–24, 1998.

[26] Bryan M. Web sgml and html 4.0 explained. *SGML and HTML Explained*, 1997.

[27] Cao J., Sun L., and Wang H. Towards secure xml document with usage control. In *Proceedings of The 7th Asia Pacific Web Conference*, volume 3399, pages 296–307, 2005.

[28] Ramaswamy Chandramouli. Application of xml tools for enterprise-wide rbac implementation tasks. In *RBAC '00: Proceedings of the fifth ACM workshop on Role-based access control*, pages 11–18, New York, NY, USA, 2000. ACM.

[29] M. Chatvichienchai, S. Iwaihara and Y. Kambayashi. Translating content-based authorizations for xml documents. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003*, pages 103– 112, 2003.

[30] Chaum D. and Van Antwerpen H. Undeniable signatures. pages 212–216.

[31] Chen F. and Sandhu R. Constraints for role-based access control. In *Proc. First ACM Workshop on Role-Based Access Control*, pages 39–46. citeseer.nj.nec.com/chen95constraints.html, 1995.

[32] Cisco Systems. E-learning. In *Partner E-Learning Connection*, 2004.

[33] Clark J. *Comparison of SGML and XML (1.0)*. World Wide Web Consortium (W3C), 1997, http://www.w3.org/TR/NOTE-sgml-xml-971215.

[34] Clark J. et al.. XML Path Language (XPath) Version 1.0. *World Wide Web Consortium (W3C)*, November 1999, http://www.w3c.org/TR/xpath.

[35] Cristina Nita-Rotaru and Ninghui Li. A framework for role-based access control in group communication systems. In *Proceedings of the International Workshop on Security in Parallel and Distributed Systems*, pages 113–128. PDCS-2004, San Francisco, 2004.

[36] Cynthia D. Copyright? protection?, the mathematics of information coding, extraction, and distribution. pages 31–47, 1999.

[37] Damiani E., Capitani S. and Samarati P. Towards securing xml web services. In *Proc. of the 2002 ACM Workshop on XML Security*, Washington, DC, USA, November 2002.

[38] Damiani E, Paraboschi S. and Samarati P. A fine-grained access control system for xml documents. *ACM Trans. Inf. Syst. Secur.*, 5(2):169–202, 2002, http://doi.acm.org/10.1145/505586.505590.

[39] Damiani E., Sabrina D., Paraboschi S. and Samarati P. Fine grained access control for soap e-services. In *Proceedings of the tenth international conference on World Wide Web*, pages 504–513. ACM Press, 2001.

[40] Damiani E., Samarati S., Vimercati D. and Paraboschi S. Controlling access to xml ddocuments. *IEEE Internet Computing*, 5(6):18–28, 2001.

[41] Damiani E., Vimercati S., Paraboschi S. and Samarati P. Design and implementation of an access control processor for xml documents. In *Computer Networks: The International Journal of Computer and Telecommunications Networking*, volume 33, pages 59 – 75. Elsevier North-Holland, New York, NY, USA, 2000.

[42] Damiani E., Vimercati S., Paraboschi S. and Samarati P. Design and implementation of an access processor for xml documents. In *Proceedings of the 9th interational WWW conference*. Amsterdam, 2000.

[43] Damiani E., Vimercati S., Paraboschi S. and Samarati P. Securing xml documents. In *Lecture Notes in Computer Science*, volume 1777, pages 121 – 135. Springer, 2000.

[44] David F. F., Dennis M. G. and Nickilyn L. An examination of federal and commercial access control policy needs. In *Proceedings of NIST NCSC National Computer Security Conference*, pages 107–116, 1993.

[45] Desmedt Y. and Frankel Y. Shared generation of authenticators and signatures. pages 457–469. CRYPTO91, 1992.

[46] Dixon W., Dawson D., Costic B. and de Queiroz M. A matlab-based control systems laboratory experience for undergraduate students: toward standardization and shared resources. *IEEE Transactions on Education*, 45(3):218–226, 2002.

[47] Edgar R. W. *Security in E-Learning*, volume 16. Springer, 2005.

[48] Edgar W. and Gerald Q. Revisiting mandatory access control: Improving the security of e-learning systems. In *Proceedings of International Conference on Communication and Computer Networks (CCN 2004)*, pages 40–45. Cambridge, IASTED, 2004.

[49] El-Khatib K., Korba L., Xu Y. and Yee G. Privacy and security in e-learning. *International Journal of Distance Education Technologies*, 1(4):11–30, 2003.

[50] Feinstein H. L. Final report: Nist small business innovative research (sbir) grant: role based access control:phase 1. *Technical report*, 1995.

[51] Ferraiolo D. F. and Kuhn D. R. Role based access control. In *Proceedings of 15th National Computer Security Conference*, pages 554–563, 1992.

[52] Finance B., Medjdoub S. and Pucheral P. The case for access control on xml relationships. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*. Bermen, Germany, 2005.

[53] Ford W. and Baum M. S. *Secure electronic commerce: Building the Infrastructure for Digital Signatures & Encryption.* Prentice Hall PTR, 1997.

[54] Freier A., Karlton P., and Kocher. P. The ssl protocol - version 3.0. 1996, http://ftp.nectec.or.th/CIE/Topics/ssldraft/INDEX.HTM.

[55] Frolik J. and Weller T.M. Wireless sensor systems: an approach for a multiuniversity design course. *IEEE Transactions on Education*, 45(2):135–141, 2002.

[56] Gabillon A. An authorization model for xml databases. In *Proceedings of the 11th ACM conference on Computer Security*, 2004.

[57] Gabillon A. A formal access control model to xml documents. In *Proceedings of the second VLDB Workshop on Secure Dta Management.* Trondheim, Norway, September 2005.

[58] Gabillon A. and Bruno E. A filtering model for xml documents. In *WWW10 Conference Workshop on Information Filtering.* Hong Kong, May 2001.

[59] Gabillon A. and Bruno E. Regulating access to xml documents. In *Proc. 15th Ann. IFIP WG 11.3 Working Conf. Database Security*, July 2001.

[60] Gandon F. and Sadeh N. Semantic web technologies to reconcile privacy and context awareness. *Web Semantics Journal*, 1(3), 2004.

[61] Gerry White. The changing landsacpe: E-learning in schools. In *Technical Report at education.au limited.* www.education.edu.au/papers/changing_landscape_gw.pdf.

[62] Gettys J., Mogul J., Frystyk H., Masinter L., Leach P. and Berners-Lee T. Hypertext transfer protocol - http/1.1. 1999, http://www.ietf.org/rfc/rfc2616.txt.

[63] Ghaoui C. and Janvier W.A. Interactive e-learning. *International Journal of Distance Education Technologies*, 2(3):23–35, 2004.

[64] Gillet D., Anh V. and Rekik Y. Collaborative web-based experimentation in flexible engineering education. *IEEE Transactions on Education*, 48(4):696–704, 2005.

[65] Goldschlag D., Reed M., and Syverson P. Onion routing for anonymous and private Internet connections. *Communications of the ACM*, 24(2):39–41, 1999.

[66] Igor T., Zachary G.I., Alon Y.H. and Daniel S.W. Updating xml. *ACM SIGMOD*, 2001.

[67] ISO. Security frameworks for open systems: Access control framework. Iso/iec 10181-3, 1996.

[68] Jajodia S., Samarati P., Subrahmanian V. and Bertino E. A unified framework for enforcing multiple access control policies. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 474–485. ACM Press, 1997.

[69] Jo S., Kim Y., Kouh H. and Yoo W. Access control model for secure xml documents. In *The Fourth Annual ACIS International Conference on Computer and Informatiom Science*. IEEE, 2005.

[70] Jothy R. and David R. *Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption.* Sams, 2004.

[71] Kakeshita T. and Murata M. A declarative manipulation for oodb and xml db having cyclic schema. 2003.

[72] Kok Kiong Tan, Tong Heng Lee and Chai Yee Soh. Internet-based monitoring of distributed control systems-an undergraduate experiment. *IEEE Transactions on Education*, 45(2):128–134, 2002.

[73] Kudo M. and Hada S. Xml document security based on provisional authorization. In *Proceedings of the 7th ACM conference on Computer and communications security*, pages 87–96. ACM Press, 2000.

[74] Kudo M. and Hada S. Xml access control. http://www.tri.ibm.com/projects/xml/xacl/xmlac-proposal.html.

[75] Kuper G., Massacci F., and Rassadko N. Generalized xml security views. In *Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 77–84. ACM Press, 2005.

[76] Lee C. S., Tan D. T. and Goh W. S. The next generation of e-learning: Strategies for media rich online teaching and engaged learning. *International Journal of Distance Education Technologies*, 2(4):1–17, 2004.

[77] Li Q. and Atluri V. Concept-level access control for the semantic web. In *Proceedings of the 2003 ACM workshop on XML security*, pages 94–103. ACM Press, 2003.

[78] Libby D. Rss 0.91 spec, revision 3. 1999, http://web.archive.org/web/20001204093600/my.netscape.com/publish/formats/rss-spec-0.91.html.

[79] Lim C.H., Park S. and Son S.H. Access control of xml documents cosidering update operations. *ACM Workshop on XML Security*, 2003.

[80] Lowe H., Wallis A. and Newman J. Role-based access control for vicarious learning. In *Proc European Conference on E-Learning*, pages 43–50. Brunel University, Uxbridge, 2002.

[81] Lu H. Open multi-agent systems for collaborative web-based learning. *International Journal of Distance Education Technologies*, 2(2):36–45, 2004.

[82] Lunt T. F. Access control policies for database systems. *Database Security, II:Status and Prospects*, pages 41–52, 1989.

[83] Lytras M. E-learn is about business: A business model for the digital economy. In *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2002*, pages 2658–2659. Montreal, Canada, 2002.

[84] Maliniak D. Design teams collaborate using internet fast track. *Electronic Design*, 49, 2001.

[85] Marilyn Greenstein and Todd M Feinman. *Electronic Commerce: Security, Risk Management and Control.* McGraw-Hill, USA, 1999.

[86] Mendling J., Neumann G., Pinterits A., Simon B. and Wild F. Indirect revenue models for e-learning at universities - the case of learn at wu. In *E-Learning Workshop, Hannover, Germany*, September 2004.

[87] Mendling J., Neumann G., Pinterits A., Simon B. and Wild F. *E-Learning, Einsatzkonzepte and Gesch Ndftsmodelle.* Physica-Verlag, Hannover, Germany, August 2005.

[88] Michael H. *XSLT Programmer's Reference.* Wiley, 2001.

[89] Miliszewska I. and Horwood J. An architecture for a federated education system. *International Journal of Distance Education Technologies*, 3(1):97–106, 2005.

[90] Murata M., Tozawa A., Kudo M. and Hada H. A framework for publishing relational data in xml. *TODS*, 27(4), 2002.

[91] Murata M., Tozawa A., Kudo M. and Hada S. Xml access control using static analysis. In *ACM Transactions on Information and System Security*, volume 9, pages 292–324. ACM Press, August 2006.

[92] Naedele M. Standards for XML and Web Services Security. *ABB Corporate Research*, 2003.

[93] Neely S., Lowe H., Eyers D., Bacon J., Newman, J. and Gong X. An architecture for supporting vicarious learning in a distributed environment. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 963–970, New York, NY, USA, 2004. ACM Press.

[94] Osborn S. L., Reid L. K. and Wesson G. J. On the interaction between role-based access control and relational databases. In *Proceedings of IFIP WG11.3 Tenth Annual Working Conference on Database Security*, pages 139–151, 1996.

[95] Park J. and Sandhu R. Towards usage control models: beyond traditional access control. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 57–64. ACM Press, 2002.

[96] Park J., Sandhu R. and Schifalacqua J. Security architectures for controlled digital information dissemination. In *Proceedings of the 16th Annual Computer Security Applications Conference*, page 224. IEEE Computer Society, 2000.

[97] Rabin M. *Digital Signatures, Foundations of secure communication.* New York: Academic Press, NewYork, 1978.

[98] Rabitti F., Bertino E., Kim W. and Woelk D. A model of authorization for next-generation database systems. *ACM Trans Database system*, 16(1):88–131, 1991.

[99] Roshan K.T. and Ravi S.S. Discretionary access control in object-oriented databases issues and research directions. In *Proc. Of the 16th NIST_NCSC National Computer Security conference*, pages 63–74, 1993.

[100] Rutgers Security Team. WWW Security. 1999, http://www-ns.rutgers.edu/www-security/.

[101] Sabrina D. An authorization model for temporal xml documents. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 1088–1093. ACM Press, 2002.

[102] Sampemane G., Naldurg P. and Campbell R. Access control for active spaces. In *ACSAC '02: Proceedings of the 18th Annual Computer Security Applications Conference*, page 343, Washington, DC, USA, 2002. IEEE Computer Society.

[103] Sandhu R. Role activation hierarchies. In *Proceedings of third ACM Workshop on RoleBased Access Control*, pages 33–40. ACM Press, 1998.

[104] Sandhu R. and Bhamidipati V. An oracle implementation of the pra97 model for permission-role assignment. In *Proceedings of ACM Workshop on Role-Based Access Control*, pages 13–21, 1998.

[105] Sandhu R. and Munawer Q. The arbac99 model for administration of roles. In *the Annual Computer Security Applications Conference*, pages 229–238. ACM Press, 1999.

[106] Sandhu R. and Park J. Usage control: A vision for next generation access control. In *MMM-ACNS 2003*, pages 17–31. Springer-Verlag Berlin Heidelberg, 2003.

[107] Sandhu R., Conyne E., Feinstein H. and Youman C. Role-based access control models. In *IEEE Computer*, number 2, pages 38–47, February 1996.

[108] Schmidt T., Wippel K.G. and Furst K. Security system for distributed business applications. *International Journal of Web Services Research*, 2(1):77–88, 2005.

[109] Seufert S. E-learning business models: Framework and best practice examples. *Idea Group*, pages 11–36, 2001.

[110] Seufert S. Lechner U. and Stanoevska K. A reference model for online learning communities. *International Journal on E-Learning*, 1(1):43–54, 2002.

[111] Shirland L.E. and Manock J.C. Collaborative teaching of integrated product development: a case study. *IEEE Transactions on Education*, 43(3):343–348, 2000.

[112] Simon E., Madsen P. and Adams C. An Introduction to XML Digital Signature. 08 August 2001, http://www.xml.com/pub/a/2001/08/08/xmldsig.html.

[113] Stafford T.F. Understanding motivations for internet use in distance education. *IEEE Transactions on Education*, 48(2):301–306, 2005.

[114] Sun L. and Li Y. Xml undeniable signature. In *Proceedings of International Conference on Computational Intelligence for Modelling, Control and Automation*. IEEE, 2005.

[115] Sun L., Li Y. and Wang H. M-service and its framework. In *Proceedings of 11th Asia-Pacific Conference on Communications*, pages 837–841. IEEE, 2005.

[116] Sun L., Wang H. and Li Y. Protecting disseminative information in e-learning. In *Advances in Web Based Learning - ICWL 2007*, pages 554–264. Springer, August 2007.

[117] Sun L.and Li Y. Dtd level authorization in xml documents with usage control. In *International Journal of Science & Network Security*, volume 6, pages 244–250, November 2006.

[118] Sun L.and Li Y. Xml schema in xml documents with usage control. In *International Journal of Science & Network Security*, volume 6, pages 170–177, December 2007.

[119] Sun L.and Li Y. Using usage control to access xml databases. In *International Journal of Information Systems in the Service Sector*, December 2008.

[120] Sun L.and Li Y. Xml and web service security. In *The 12th International Conference on Computer Supported Cooperative Work in Design*, volume 4823/2008, pages 765–770. IEEE, April 2008.

[121] Technology Reports. Business rules markup language. 2002, http://xml.coverpages.org/brml.html.

[122] Terry R. Application level security using an object-oriented graphical user interface. In *Proceedings on the 1992-1993 workshop on New security paradigms*, pages 105–108. ACM Press, 1993.

[123] Velev M.N. Integrating formal verification into an advanced computer architecture course. *IEEE Transactions on Education*, 48(2):216–222, 2005.

[124] W3C. Mathematical markup language (mathml) 1.01 specification. 1999, http://www.w3.org/TR/REC-MathML/.

[125] W3C. Xml pointer language (xpointer) version 1.0. 2001, http://www.w3.org/TR/REC-html40/.

[126] W3C. Xml linking language (xlink) version 1.0. 2001, http://www.w3.org/TR/xlink/.

[127] W3C. Xhtml 1.0 the extensible hypertext markup language (second edition). 2002, http://www.w3.org/TR/xhtml1/.

[128] W3C. Rdf/xml syntax specification. 2004, http://www.w3.org/TR/rdf-syntax-grammar/.

[129] W3C. Extensible stylesheet language (xsl) version 1.1. 2006, http://www.w3.org/TR/xsl/.

[130] Jingzhu Wang and Sylvia L. Osborn. A role-based approach to access control for xml databases. In *SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 70–77, New York, NY, USA, 2004. ACM.

[131] Wang H., Cao J. and Zhang Y. A consumer anonymity scalable payment scheme with role based access control. *In: 2nd International Conference on Web Information Systems Engineering(WISE2001)*, pages 53–62, 2001.

[132] Wang H., Cao J. and Zhang Y. Formal authorization allocation approaches for role-based access control based on relational algebra operations. *In: 3nd International Conference on Web Information Systems Engineering(WISE2002)*, 2002.

[133] Wang H., Cao J. and Zhang Y. Formal authorization allocation approaches for permission-role assignments using relational algebra operations. In *Proceedings of the 14th Australian Database Conference ADC2003*, Adelaide, Australia, 2003.

[134] Wang H., Cao J., Zhang Y. A flexible payment scheme and its role based access control. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):425–436, 2005.

[135] Wang H., Sun L., Cao J., and Zhang Y. Anonymous access scheme for electronic-services . In *Proceedings of the Twenty-Seventh Australasian Computer Science Conference (ACSC2004)*, pages 296–305, Dunedin, New Zealand, 2004.

[136] Wang H., Sun L., Zhang Y., and Cao J. Authorization Algorithms for the Mobility of User-Role Relationship. In *Proceedings of the 28th Australasian*

Computer Science Conference (ACSC2005), pages 167–176, Newcastle, Australia, 2005.

[137] Wang H., Zhang Y., Cao J. and Varadharajan V. Achieving secure and flexible m-services through tickets. *IEEE Transactions on Systems, Special issue on M-Services*, 33:697–708, 2003.

[138] Wang Y. and Tan K. A scalable xml access control system. In *Proceedings of the 10th interational WWW conference*. Poster, 2001.

[139] William H. Graves. The new challenges of e-learning. *Ubiquity*, 1(43):2, 2001.

[140] Zhang G. and Parashar M. Context-aware dynamic access control for pervasive applications. In *CND'04: Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference*, pages 21–30. Society for Modeling and Simulation International, 2004.

[141] Zhang X., Park J. and Parisi-Presicce F. A logical specification for usage control. In *SACMAT'4*. ACM Press, 2004.

[142] Zhang X., Park J. and Sandhu R. Schema based xml security: Rbac approach. In *Proceedings of the IFIP WG*. ACM Press, 2003.