Securing Financial XML Transactions Using Intelligent Fuzzy Classification Techniques

A Smart Fuzzy-based Model for Financial XML Transactions Security using XML Encryption

By

Faisal Tawfiq Ammari

Submitted for the degree of Doctor of Philosophy

School of Computing and Engineering

University of Huddersfield

October 2013

SECURING FINANCIAL XML TRANSACTIONS USING INTELLIGENT FUZZY CLASSIFICATION METHODS

By

Faisal Tawfiq Ammari

ABSTRACT

***Keywords*:** XML, financial transactions, fuzzy logic, fuzzy XML, XML security, XML encryption, financial transactions, banking

The eXtensible Markup Language (XML) has been widely adopted in many financial institutions in their daily transactions; this adoption was due to the flexible nature of XML providing a common syntax for systems messaging in general and in financial messaging in specific. Excessive use of XML in financial transactions messaging created an aligned interest in security protocols integrated into XML solutions in order to protect exchanged XML messages in an efficient yet powerful mechanism.
However, financial institutions (i.e. banks) perform large volume of transactions on daily basis which require securing XML messages on large scale. Securing large volume of messages will result performance and resource issues. Therefore, an approach is needed to secure specified portions of an XML document, syntax and processing rules for representing secured parts.

In this research we have developed a smart approach for securing financial XML transactions using effective and intelligent fuzzy classification techniques. Our approach defines the process of classifying XML content using a set of fuzzy variables. Upon fuzzy classification phase, a unique value is assigned to a defined attribute named "Importance Level". Assigned value indicates the data sensitivity for each XML tag.

This thesis also defines the process of securing classified financial XML message content by performing element-wise XML encryption on selected parts defined in fuzzy classification phase. Element-wise encryption is performed using symmetric encryption using AES algorithm with different key sizes. Key size of 128-bit is being used on tags

classified with "Medium" importance level; a key size of 256-bit is being used on tags classified with "High" importance level.
An implementation has been performed on a real-life environment using online banking system in Jordan Ahli Bank one of the leading banks in Jordan to demonstrate its flexibility, feasibility, and efficiency. Our experimental results of the system verified tangible enhancements in encryption efficiency, processing-time reduction, and resulting XML message sizes.

Finally, our proposed system was designed, developed, and evaluated using a live data extracted from an internet banking service in one of the leading banks in Jordan. The results obtained from our experiments are promising, showing that our model can provide an effective yet resilient support for financial systems to secure exchanged financial XML messages.

ACKNOWLEDGMENT

"*I can do all things through him who strengthens me*". First of all, I would like to thank Christ, who made me believe in myself to complete this work and reach this important stage of my life to fulfil my dream.

I would like to express my appreciation to my supervisor, Professor Joan Lu, for her continuous guidance and great support through this journey. She showed a great dedication and commitment providing a provisional and exceptional support and encouragement to shed the light through my research.

Great appreciations go to Dr. Maher Aburrous for being a real support guiding me through this research. I am really grateful for all his paid efforts and support.

Finally, my gratitude goes to the University of Huddersfield, providing the opportunity to pursue my dream and complete my degree.

DEDICATION

To my ***beautiful wife***, my partner, love of my life (Salam), who never stopped dripping the love from her heart to mine, who never stopped encouraging me, and who never stopped lighting my way through the journey towards success.

To my ***wonderful parents*** (Tawfiq and Amal), who never stopped believing in me, who never stopped supporting and loving me, and who never stopped drawing the smile on my face all the time during my long journey.

To ***My beloved Brothers*** (Tariq, Salem, Sultan, Waleed), who never stopped being by my side, who never stopped pushing me when needed, and who never stopped believing in me.

To my Friends and Relatives, thank you all.

List of Acronyms and Abbreviations

AES	Advanced Encryption Standard
AI	Artificial Intelligence
API	Application Program Interfaces
CBA	Classification based on Association Rule
COG	Centre of Gravity
DES	Data Encryption Standard
DOM	Document Object Model
DRM	Digital Rights Management
DSD	Document Structure Description
DTD	Document Type Definition
FL	Fuzzy Logic
IDE	Integrated development environment
IDEA	International Data Encryption Algorithm
KDC	Key Distribution Centre
KEK	Key Encryption Key
kNN	k-nearest Neighbour Classification
LLSF	Linear Least Squares Fit Mapping
OASIS	Organization for the Advancement of Structured Information Standards
PAP	Policy Access Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point

PKI	Public Key Infrastructure
SAML	Security Assertion Markup Language
SAX	Simple Application Interface for XML
SVMs	Support Vector Machines
SXMS	Secure XML Management System
TEK	Traffic Encryption Key
TTP	Trusted Third Party
UML	Unified Modelling Language
URI	Uniform Resource Identifier
X-KISS	XML Key Information Service Specification
X-KRSS	XML Key Registration Service Specification
XAC	XML access control
XACML	XML Access Control Mark-up Language
XEnc	XML Encryption Syntax and Processing
XKMS	XML key management specification
XML	eXtensible Markup Language
XNI	Xerces Native Interface
XSL	eXtensible Stylesheet Language
XSLT	XSL Transformations
W3C	World Wide Web Consortium

Contents

List of Figures

List of Tables

List of Publications

Journal Papers

1. Ammari, F., Lu, J., Aburrous, M., "Intelligent Banking XML Encryption using Effective Fuzzy Classification", Book Chapter - Emerging Trends in Information and Communication Technologies Security, ISBN 9780124114746 (Print), Elsevier, September-October 2013.
2. Ammari, F., Lu, J., Aburrous, M., "Enhanced XML Encryption Using Classification mining techniques for e-Banking transactions", (Submitted)

Conference Papers

1. Ammari, F., Lu, J., "Improved Banking XML Transaction Encryption Using Tag Fuzzy Classification," *Proceeding of International Conference on Computer Applications Technology*, Sousse, Tunisia, January 20-22 2013.
2. Ammari, F., Lu, J., Aburrous, M., "Intelligent XML Tag Classification Techniques for XML Encryption Improvement", *Proceeding of the 3rd IEEE International Conference on Privacy, Security, Risk and Trust*, MIT, Boston, USA October 9-11, 2011.
3. Ammari, F., Lu, J., Aburrous, M., " Advanced XML Security: Framework for Building Secure XML Management System (SXMS)"**,** *Proceeding of the 7th International Conference on New Generations (ITNG), Las Vegas, USA*, April, Page(s): 120 – 125, 2010

Chapter 1

Introduction

1.1 Overview

eXtensible Markup Language (XML) (Bray, Paoli, Sperberg-McQueen, Maler, & Yergeau, 2008) has been widely adopted in many financial institutions in their daily transactions; this adoption has been due to the flexible nature of XML in providing a common syntax for systems messaging in general and for financial messaging in particular. Excessive use of XML in financial transactions messaging has created an aligned interest in security protocols integrated into XML solutions in order to protect exchanged XML messages by using an efficient yet powerful mechanism. There have been several approaches proposed by researchers to secure XML messages and there is a comprehensive collection of related works.

XML is designed based on text format and has a tree structure. It is natural that data integrity, data authentication, information confidentiality, and other security benefits

should be applied to entire XML data or portions of XML data. XML security solutions should provide a high level of security to ensure the confidentiality of information represented using the XML format. XML security must be integrated with XML data features and characteristics to keep the flexible nature of XML while integrating essential security technologies.

Due to the sensitive nature of financial transactions that use XML as their main messaging protocol, a security requirement should be fulfilled to protect exchanged XML messages by using a dynamic and efficient mechanism. The security mechanism should encrypt portions of XML data rather than whole messages, e.g. element-wise encryption should be used to protect sensitive parts within the XML message.

The specifications related to XML security published by W3C define the basic framework and rules that can be utilized across applications. The basic idea for XML security is to perform data encryption on XML messages whereby XML data confidentiality is achieved to ensure that the XML data structure, data content, and other sensitive information in XML data may only be accessed by legitimate parties. Confidentiality is generally associated with encryption mechanisms or access control technologies. XML key management (Hallam-Baker & Mysore, 2005) provides the basic key requirements for XML data confidentiality .

However, on a daily basis, financial institutions (i.e. banks) perform large volumes of transactions that require XML encryption on a large scale. Encrypting large volumes of messages in full will result in performance and resource issues. Therefore, an approach is needed to encrypt defined parts within the XML document, to identify syntax for representing encrypted portions, and to identify the processing rules for decrypting those portions. W3C XML encryption has a feature called element-wise encryption,

which is the process of encrypting parts of an XML document. The encryption process can be applied to more than one element in a given XML document; each is contained in another element. The element might enclose sub-elements, attributes, texts, or a mix of all mentioned items. The remaining parts of the document should remain intact as plaintext.

To avoid any performance or resource issues, a mechanism should be considered to choose which parts of the XML document should be encrypted on the fly, whereby the parts are selected based on smart criteria for detecting sensitive information within an XML document .

The fuzzy logic (FL) (L.A. Zadeh, 1965) approach can be used to distinguish sensitive parts within each XML document. FL provides an easy way to reach to a definite conclusion based upon noisy, vague, imprecise, ambiguous, or missing information. FL's approach for controlling problems imitates how a person would make a quick decision. FL includes a rule-based 'IF X AND Y, THEN Z' approach for solving a control problem, rather than attempting to design a system in mathematical way. The FL model is relying on an operator's experience rather than their technical understanding of the system.

The FL approach is quantified based on a combination of historical data and expert input. FL has been used in many fields especially in computer information systems, and computer science to combine expert input with computer models for a large scale of applications. The main advantage of the fuzzy approach is that it can process imprecisely defined variables and variables which mathematical relationships cannot define their corresponding relationships. FL has the ability to integrate expert human knowledge and judgement to define the variables and corresponding relationships. By integrating expert human judgement we get more realistic model (Mahant, 2004)

1.2 Research Motivation

Many businesses and financial institutions use XML in their basic transaction messaging, due to its flexible nature and structure. A solid security approach is required to ensure safe and trustworthy transactions either within the same institution or between different institutions (business to business). XML security specifications published by W3C have addressed XML encryption (Imamura, Dillaway, & Simon, 2002). XML encryption is mainly used to ensure XML data confidentiality and authenticity. However, institutions that deal with large volumes of transactions on a daily basis require a flexible and solid mechanism to process XML messages that performs element-wise encryption in a timely and efficient manner .

Encrypting sensitive information only within the XML document is a complicated issue to analyse; it needs to take into consideration the process of encrypting different portions within the XML document every time the message is transmitted. This is complex to analyse because the classification requires set of factors to consider.

Despite there are handful of applications available to secure XML messages, there are no known solutions that utilize fuzzy classification techniques in detecting sensitive information within each XML document.

The motivation behind our research is to create an effective, powerful, and intelligent model to classify XML messages by detecting sensitive information within XML documents, in order to perform element-wise encryption on selected parts .

We have conducted a quantitative methodology in our research, and it explores fuzzy XML classification systems. The technique uses FL to process XML data features and patterns, for extracting the message's fuzzy classification rules into the data miner, and

then for applying element-wise encryption algorithms on selected portions extracted from XML data features and patterns. The proposed XML security model combines fuzzy techniques for the purpose of automating the fuzzy rules. The automation process is completed by extracting the set of fuzzy rules that are going to be deployed inside the fuzzy inference engine. Therefore, a set of IF-THEN rules are constructed using these fuzzy rules. The IF-THEN rules reflect the relations between different transaction characteristics and patterns and their associations with one another. These rules can then be used for the final stage, which uses element-wise encryption with different key sizes based on the importance levels assigned .

Our previous expertise in the banking and financial sector in Jordan shed the light on the importance of securing business and financial transactions based on XML. Many researches and case studies have been reviewed and evaluated in order to find a robust and flexible solution that can be used with ease and efficiency.

1.3 Aims & Objectives

Our prime aim is to build a system that secure financial XML messages that uses FL to classify XML content to perform XML encryption, in order to provide necessary data confidentiality for classified portions within an XML document. This mechanism enhances the performance of encrypting XML documents on high-volume transactions. In order to reach this aim, there are several objectives:

Our objectives can be summarized with the following points:

1. Conduct a literature review to illustrate the current and existing approaches concerning XML security, fuzzy XML models, and XML data confidentiality.

2. Build a resilient, intelligent, dynamic, and secure XML messaging system that uses artificial intelligence and FL to fetch and classify sensitive information within XML documents. The outcome system has to be adaptive, flexible, and efficient.
3. Illustrate feasibility and adaptability after analysing a large number of actual transaction datasets fetched over period of time reflecting internet banking transactions, phone banking transactions, and mobile banking transactions.
4. Present the applicability of applying an FL expert system in order to classify and find out importance levels within XML documents. Fuzzy rules are being used for and driven by human expert knowledge on creating flexible and secure XML messaging systems.
5. Provide a solution that improves existing XML encryption approaches and illustrates a performance improvement over well-known XML encryption models, like W3C XML Encryption Recommendations. We illustrate the enhancements and processing improvements by securing only the necessary parts with high importance levels within XML documents.

We have implemented a quantitative research methodology to enable us achieving all above objectives, taking into consideration experimental studies, case analysis, data collection, testing, evaluation, and comparing final results.

1.4 Research Contribution

This research will contribute to the fields of XML security, fuzzy XML in general, XML encryption, and fuzzification, specifically in the following areas:

1. A robust XML security model has been introduced performing element-wise encryption to encrypt critical and sensitive parts within XML documents that are

used in business and financial institutions. This technique encrypts XML documents efficiently by encrypting only the sensitive parts within each XML document by using a set of fuzzy variables. Significant processing time improvements have been recorded against a well-known approach (W3C XML Encryption Recommendations).

2. An efficient and secure XML model has been created to reduce encrypted XML file sizes, due to the fact that only essential parts within each XML document are encrypted, leaving other nodes intact without encryption. Reduce file sizes will reduced bandwidth used on large scale.
3. A fuzzy classification engine has been created to determine which parts within each XML document require encryption. The fuzzification engine works by assigning an importance level to each tag within the XML document. The importance level value is set by the engine for later encryption processing; the importance level is assigned using one of three values (high, medium, or low). The fuzzification process itself is based on fuzzy logic, which uses a combination of human expert knowledge and a set of IF-THEN operators.
4. A resilient XML encryption system has been created whereby future enhancements can be achieved without the need to redesign the whole system. The encryption standards and algorithms used within the system are flexible and can be changed when needed; different encryption algorithms can be used to replace existing algorithms either to improve performance or to add more security.
5. A set of XML fuzzy classification characteristics has been created that were fetched from actual financial XML transactions. There are ten main characteristics that were created by using a hybrid of personal expertise, onsite

financial analysts, a set of questionnaires, and a set of surveys. These ten characteristics define the classification criteria we will use in our fuzzy classification stage.

6. A desktop application has been designed and developed to test and validate our proposed model. Application developed based on reliability, feasibility, and extraction process of the mechanism. The application was programmed using the Java programming language.

1.5 Outline of the Thesis

Here to present a brief outline of the thesis:

In Chapter 2, the research background is discussed. XML models are included, XML application program interfaces, XML schema languages, cryptographic algorithms for data confidentiality, XML security models (with XML encryption standards described), text categorization, and FL models and design.

In Chapter 3, the literature review is presented. The contents of the literature review mainly focus on existing ideas and solutions related to XML encryption. It looks at the existing models, theories, and schemes of XML encryption for ensuring XML data confidentiality. Existing fuzzy XML techniques and models are also addressed to cover all aspects of our research.

In Chapter 4, a full description is given and introduced of a novel model for securing financial XML documents by encrypting specific portions within the documents. Selection is performed based on the XML fuzzification phase to determine which parts are to be encrypted and what types of encryption algorithm and key are to be deployed. We call our model SXMS, which stands for secure XML management system. This

description includes the main model design, concepts, and workflow that are used in this model .

An illustration of the system design and architecture and the tools used for implementing the system of the SXMS model is also given. A theoretical execution based on the case study is also offered. This section presents the classes for encrypting and decrypting XML messages based on the importance level assigned, as well as the process of building the final XML document by deploying the needed XML encryption with different key sizes based on XML data sensitivity for each XML portion .

In Chapter 5, a detailed system implementation has been described. Development architecture and used tools have been introduced to elaborate on how the system implemented. Used testing strategy, testing behaviour, testing data, and testing functionality have been introduced as well. Testing validation against W3C XML Recommendations (Imamura et al., 2002) has been presented to present SXMS advantage. Finally an independent testing for the major modules, fuzzy classification stage, and XML encryption has been illustrated.

In Chapter 6, the full performance evaluation stages are discussed. The evaluation stages are presented to give a clear idea of the mechanism we followed in our evaluation, presenting the basic components that we used in our evaluation and judgements. Finally, we provide screenshots of the developed application and samples of the source code used to develop this application .

In Chapter 7, an overview of the thesis along with a discussion of future research directions are presented.

Chapter 2

Research Background

2.1 Introduction

In this chapter we introduce the XML model with all its detailed components, and then we briefly introduce the XML language and its structure. Cryptographic algorithms and types are presented. The XML security model is presented as well. Specifically, we present XML digital signatures, XML Encryption, XML key management, SAML, and XACML security models. We then present the information retrieval mechanism in general and XML fuzzy classification specifically. Finally, we introduce the fuzzy logic model in details.

2.2 XML Model

EXtensible Markup Language (XML) was first announced in 1998 (W3C, 1998). This proposed markup language became the standard for data exchange and representation among many online and offline applications, providing the flexibility of exchanging different digital content among applications. An XML document is formed in a hierarchical structure with the ability to define its element contents, define tags, create

nested document structures, and build document types by specifying a set of regular expression patterns (XML Schema or Document Type Definitions (DTD)). An XML document incorporates structure and data in one entity. Therefore, XML data is semi-structured data (S. Abiteboul, 1996).

The solid set of XML characteristics created an interest in building effective solutions to support the following advantages:

- Extensibility: New fields and tags can be created when they are needed. There are no fixed set of fields.
- Self-description: This feature allows any XML field to process an unlimited number of attributes.
- Readability: XML is easy for humans or machines to read and understand. This feature facilitates the usage of XML by different applications and users.
- Simplicity: XML code is easy to understand; also, it can be easily processed and deployed in different practices. Updating the existing XML Schema is an easy and straightforward operation.
- Supports multilingual documents and Unicode: This is important for the internationalization of applications.
- Interoperability: There is the ability to use XML documents in any industry without the need to make changes to the data itself. XML is treated as an independent unit from both the machine and software levels.
- Portability: An XML document has the ability to represent different data types, such as ordinary text and binary files (images, videos, and sounds).

However, there are some drawbacks in XML, which include:

- Syntax redundancy: This can affect human readability and system efficiency, and this can result in higher storage requirements and resource usage. Bandwidth limitations can prevent XML being deployed in certain applications.
- A number of vague, unneeded features within XML: Efforts were made to create "Minimal XML", which led to the discovery that there was no consensus on which features were in fact obscure or unnecessary.
- A wide range of data types are not supported in the basic parsing requirements: Some additional work might be required to process the desired data in the XML document.
- A significant overhead for various uses of XML: This mainly applies where resources may be limited. This might happen because of the parser's limitations in recursing arbitrarily nested data structures or the missing feature of performing additional checking and validation for improperly formatted syntax or data.
- Security concerns: These may arise when XML input is fed from unknown or untrusted sources
- Difficulty in modelling overlapping data structures: This requires extra effort.

2.2.1 Well-formed XML

A well-formed XML document is one which corresponds to the XML 1.0 (Bray, Paoli, Sperberg-McQueen, Maler, & Yergeau, 2007) grammar specified by W3C. It has just one root element, which is known as the document element. Each starting element tag must have a corresponding closing tag. Each element ought to be nested within one another. Nesting rules with defined labels enable information to be represented by XML hierarchically. Figure 2.1 illustrates a sample of a well-formed XML message.

```
<?xml version = "1.0" encoding="UTF-8" standalone="yes"?>
<document>
  <employee>
    <name>
      <lastname>Faisal</lastname>
      <firstname>Ammari</firstname>
    </name>
    <hiredate>January 2, 2000</hiredate>
      <project>
        <product>Laptop</product>
        <id>002</id>
        <price>JD500.125</price>
      </project>
    </projects>
  </employee>
</document>
```

Figure 2.1: Sample of a well-formed XML message

2.2.2 XML Components

An XML document consists of elements that represent a piece of information. More specific information can be found in nested elements, such as character data, attributes, and entity references. These elements are marked up by the tags in a specific document. Between the start tag and end tag of an element there is the element content, presented as text content.

1. Elements: Elements start with an opening tag (<t>) and end with an ending tag (</t>). Everything between the starting and ending tags is called the element content. Each element has an element name (e), which should follow the following rules:
   - The element name should not start with "XML" or "xml".
   - The element name is case sensitive.
   - The element name starts with a character or an underscore.
   - The element name consists of characters, numerals, underscores, and tabs.
2. Attributes: Attributes are information that can provide more information about the element and often define an instance of an element. Attributes have a name

defined. For example, <book name="Learn C++ in 24 days"> is a book element with an attribute name that has the value "Learn C++ in 24 days".

3. Comments: Comments can be placed anywhere inside the XML document for further explanation or description. Comments are not part of the main document and can be used by using the start tag (<!—) and the end tag (-->).
4. XML declaration: An XML declaration supplies the XML processor with information such as encoding, version, and any other information related to the document. A declaration can be defined with a start tag (<?xml) and an end tag (?>).
5. Processing Instructions (PIs): PIs may occur anywhere in the XML document. Their main purpose is to carry specific instructions to the application. A PI is represented within the document in the form of : <?Target instructions for command?>.
6. CDATA sections: Using CDATA will tell the XML parser that there is no markup in the characters during the time of processing. Sections can be defined by using: <! [CDATA]!>.

2.3 XML Schema Languages

XML schema languages, like XML Schema (D. C. Fallside & Walmsley, 2004), DTDs, DSDs (M\oller, 2005), Schematron (Jelliffe, 2006), and RELAX NG (Makoto, Walsh, & McRae, 2001), are used to certify XML documents. The reason for certifying/validating a document is to verify whether the XML document conforms to a set of structural and content rules expressed in one of many schema languages. The validation procedure happens on at least four primary levels (Ray, 2003):

1. Structure: Relates to the placement and use of markup elements and attributes.

2. Data typing: Relates to the set of numbers, dates, and texts (patterns of character data).
3. Integrity: Relates to the linkage between resources and corresponding nodes.
4. Business rules: Relates to collections of tests such as spelling checks, checksum results, etc.

2.3.1 Document Type Definition (DTD)

The Document Type Definition (DTD) (Hunter, Cagle, Dix, & Cable, 2001) is a set of rules that define the hierarchical structure of any XML document. The XML parser utilizes these rules to determine whether the XML document is valid or not. The DTD consists of four basic parts: elements, attributes, tags, and entities. A declared element specifies the name of the element and the valid content. Tags are used to indicate elements. Attributes are used to provide additional details about an element and can be used to describe element properties as well. Entities are the variables that can be reused within the document. Figure 2.2 illustrates a sample DTD example.

```
<!DOCTYPE family [
   <!ELEMENT title (#PCDATA)>
   <!ELEMENT parent (#PCDATA)>
   <!ATTLIST parent role (mother | father) #required>
   <!ELEMENT child (#PCDATA)>
   <!ATTLIST child role (daughter | son) #required>
   <!NOTATION gif system "image/gif">
   <!ENTITY JENN system
   "http://images.about.com/sites/guidepics/html.gif"
   NDATA gif>
   <!ELEMENT image empty>
   <!ATTLIST image source entity #required>
   <!ENTITY footer "Brought to you by Jennifer Kyrnin">
]>
```

Figure 2.2: A DTD Example

2.3.2 XML Schema

XML Schema (Hunter et al., 2001) is a DTD alternative that is based on XML. Users can exploit XML Schema to represent an XML document structure. An XML Schema Definition (XSD) is regarded as XML Schema language.

XML Schemas can take the place of DTDs in most web applications. XML Schemas are written in XML and are present in well-formed XML documents. XML Schemas support data types such as string, date and time, and integers. XML Schemas can be used to construct complex data types as well.

2.3.3 RELAX NG

RELAX NG is a schema language for XML developed by ISO/IEC JTC1/SC34/WG1 (Makoto et al., 2001). RELAX NG is based on two languages. The first language is the Tree Regular Expressions for XML (TREX) designed by James Clark (Clark, 2001); the second language is the Regular Language description for XML (RELAX) designed by Murata Makoto (Makoto, 2002). The primary idea of RELAX NG consists of patterns that are formed to widen the range of the idea of the content model. In RELAX NG, a pattern is an expression of elements, text nodes, and attributes. The definitions of data types may be utilized for constraining the sets of values of text nodes and attributes. Figure 2.3 demonstrates an example of a RELAX NG Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<element name="books"
xmlns="http://relaxng.org/ns/structure/1.0">
   <element name="book"
   <attribute name="price"/>
   <attribute name="id"/>
   <element name="name"><text/></element>
   <element name="authors"
      <element name="author"><text/>
      <attribute name="id"/>
      <optional>
        <attribute name="address"/>
      </optional>
      </element>
   </element>
   <element name="ISBN"><text/></element>
   </element>
</element>
```

Figure 2.3: A RELAX NG Schema example

2.3.4 Document Structure Description (DSD)

A Document Structure Description is a schema language developed by combined efforts from BRICS and AT&T Labs (Klarlund, Møller, & Schwartzbach, 2000; M\oller, 2005). Constraints form the basic concept of a DSD. A restraint is used to define the content of an element, corresponding attributes, and its particular context. A pair consisting of an element name and a restraint forms the definition of an element. Element content is constrained by means of a content expression, which is a regular expression over element definitions. To force constraints on the context of an element, context patterns are used.

2.3.5 Schematron

Schematron is schema language based on rules developed by Rick Jelliffe at the Academia Sinica Computing Centre (ASCC) (Jelliffe, 2006). The main functionality of Schematron is to perform co-constraints checking in XML instance documents. A sequence of rules is defined by a Schematron document, which is grouped in a logical way in pattern elements. A context attribute is included within each rule and used by an XPath pattern to determine the elements to which the rule applies. Within each rule, a sequence of reports and elements are specified with a testing attribute, which considered an XPath expression. This expression is evaluated to a value of Boolean type for each node within the context.

2.4 XML API

Application Program Interfaces (APIs) are used by many programming languages to access XML document information without the need to create and write a parser in the specific used language.

2.4.1 DOM Parser

The DOM (Document Object Model) parser is utilized as a hierarchical object model to get XML document information (Hégaret, Whitmer, & Wood, 2005). The whole XML document's information is being read by DOM parser, and establishes the corresponding DOM object tree of nodes. The construction is being performed in main memory. This XML parser is appropriate for small to medium XML documents that may fit in memory. DOM can be utilized in document-centric document whereby the sequence of elements within the document is essential. The sequence of elements is preserved due to direct read from the document. It has multiple functions for traversing XML trees and other relevant functions like (insert, delete, and access nodes).

2.4.2 SAX Parser

The Simple Application Interface for XML (SAX) parser provides accessibility to XML information as a series of events. For every open tag, closing tag, and every #PCDATA and CDATA section, SAX activates an event. These events along with the corresponding sequences need to be interpreted by the document handler. SAX is suitable for medium to large XML documents; this is because there is no need to parse XML documents in main memory first. SAX is suitable for structured XML documents as well, as the sequence for elements not essential.

2.4.3 Java API for XML

The Java programming language (and some other languages) provides different types of XML Application Programming Interfaces (APIs), such as SAX, DOM, and XSLT (Mclaughlin & Edelson, 2006; Violleau, 2001; Williams, 2009), in order to process XML documents by means of writing a computer program using several programming

languages. SAX (Simple API for XML) scans the XML document sequentially and throws up events that the programmer can handle. These events are thrown up by the parser when it detects the start document and end document tags, as well as the start element tag, including a list of all its attributes, end elements, and characters. The programmer should write suitable code for each event to process an entire XML document. Since each event occurs only once for each element, all the required work needed to process the document should be done in one cycle.

2.5 Cryptographic Algorithms for Confidentiality

2.5.1 Encryption Mechanism

The encryption mechanism is the process of transforming plaintext into ciphertext and vice versa. Plaintext is readable content that needs rendering to be unreadable. Ciphertext is encrypted plaintext in which no semantic content is available. An encryption mechanism transforms plaintext into ciphertext. A decryption mechanism transforms ciphertext back into plaintext. In order to be able to re-use a particular algorithm in many systems, the algorithm is parameterized by a key. Figure 2.4 illustrates encryption mechanisms.

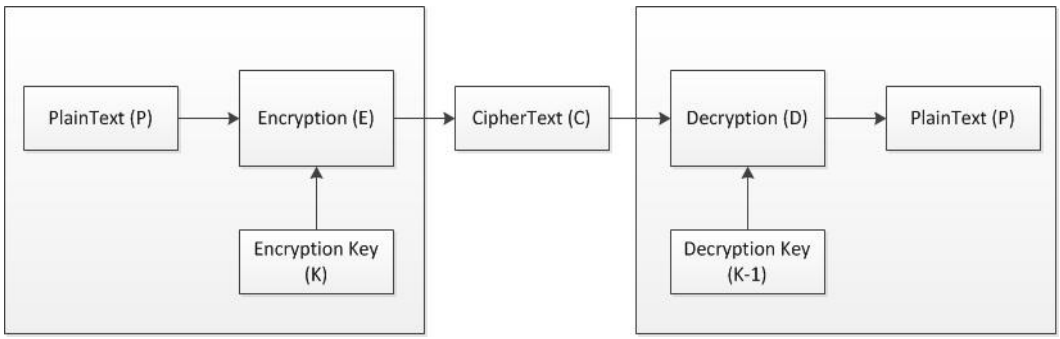

Figure 2.4: A generic encryption system

Key: "A sequence of symbols that controls the operations of encryption and decryption."

As illustrated in Figure 2.4, to encrypt plaintext (P), the encryption algorithm (E) is parameterized with a confidentiality encryption key (K). Ciphertext (C) is decrypted using the confidentiality decryption key (K^{-1}).

Encrypting plaintext (P) under the key (K) produces ciphertext (C) and is denoted as$C = E_K(P)$. The decryption of C under the key K^{-1} reproduces the plaintext P and is denoted as $P = E_{K^{-1}}(C)$.

Cryptography uses two different types of encryption systems: symmetric encryption systems and asymmetric encryption systems.

2.5.2 Symmetric Cryptography

Symmetric encryption system: This type of encryption system is based on symmetric cryptographic techniques that use one secret key for both the encryption and decryption algorithms.

Symmetric cryptographic technique: This cryptographic technique uses a shared secret key. Symmetric encryption systems have the property that both the encryption key and the decryption key have the same value.

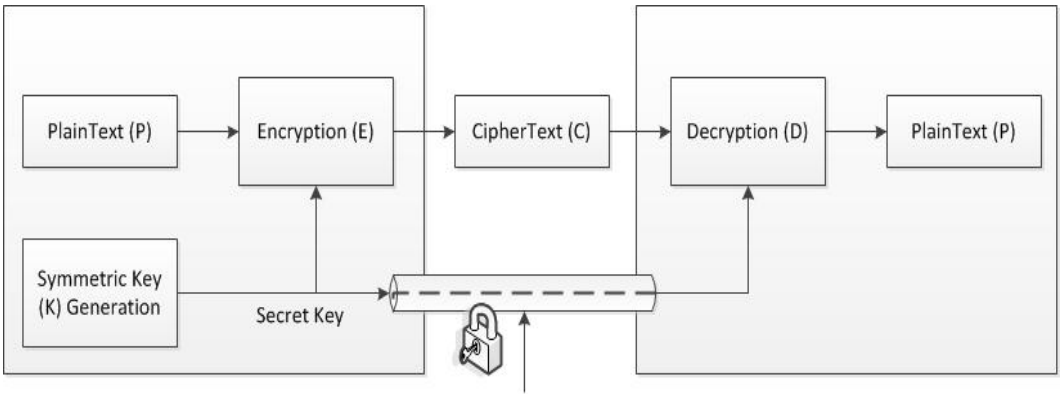

Figure 2.5: Symmetric cryptographic cycle

Secret Key: “A key that is used with a symmetric cryptographic algorithm. Possession of a secret key is restricted (usually to two entities)” (ISO10181-1). The secret key can be generated in various ways:

- by the encrypting entity (encryptor) illustrated in figure 2.5;
- by the decryptor;
- by a trusted third party (TTP), like a key distribution centre (KDC); or
- It can be derived from parameters in a key agreement protocol that is performed by both the encryptor and the decryptor.

If the key is not computed using a key agreement protocol, the key must be transported through a secure channel that is protected in terms of confidentiality and integrity. Additionally, the recipient(s) of the secret key must know the source of the key, i.e. data origin authentication for the transported key is necessary.
The ciphertext itself can be transported through an unprotected channel.

Symmetric cryptographic algorithm: “An algorithm for performing encryption or the corresponding algorithm for performing decryption in which the same key is required for both encryption and decryption” (ISO10181-1).
Algorithms that perform symmetric encryption are grouped into two classes:

1. Block ciphers
2. Stream ciphers

A block cipher processes blocks of plaintext to create blocks of ciphertext. A block is a string of n bits; such a block cipher is called an n-bit block cipher. Typical algorithms that are used in today’s systems include:

- AES (Advanced Encryption Standard)

- 3DES (Triple DES (Data Encryption Standard), also known as TDEA: Triple Data Encryption Algorithm)
- IDEA (International Data Encryption Algorithm)
- Various other block ciphers, like the other AES candidates (e.g. Blowfish or RC6).

The plaintext bit sequence is segmented into n-bit blocks, as the block cipher needs n bit as its input. To allow cases where the input length is not a multiple of n bit, a padding mechanism is usually used with a block cipher. The padding algorithm defines an unambiguous way in which each plaintext is extended to a length of a multiple of n bit. This is done even if the length of the plaintext is already a multiple of n bit. So, if a padding mechanism is used, the length of the ciphertext is larger than the length of the plaintext. After decrypting with the block cipher, the padded bits are removed from the decrypted data.

The mechanisms closely related to block ciphers are modes of operation. Modes define how the inputs and outputs of consecutive block cipher operations are combined. This is done to ensure that the same plaintext block results in different ciphertext blocks throughout the ciphertext stream and to chain the blocks together to prevent substitution attacks .

A stream cipher combines a sequence of plaintext symbols with a sequence of keystream symbols, one symbol at a time, using an invertible function (for single bits as a symbol, the function is usually an exclusive bit or a between-keystream bit and a plaintext bit). Typical stream cipher algorithms include RC4 and A5, which are n-bit block ciphers that operate in a specific mode to create a key symbol stream.

2.5.3 Asymmetric Cryptography

Asymmetric encryption system: "Encryption system based on asymmetric cryptographic techniques whose public transformation is used for encipherment and whose private transformation is used for decipherment" (ISO/IEC9798-1, 1997).

Asymmetric cryptographic technique: "Cryptographic technique that uses two related transformations, a public transformation (defined by the public key) and a private transformation (defined by the private key). The two transformations have the property that, given the public transformation, it is computationally infeasible to derive the private transformation" (ISO/IEC11770).

In an asymmetric encryption system, the encryption key K (also called the public key) and the decryption key K^(-1) (also called the private key) have distinct values, but these values have a mathematical relationship that is defined by the underlying cryptographic algorithm, as illustrated in figure 2.6

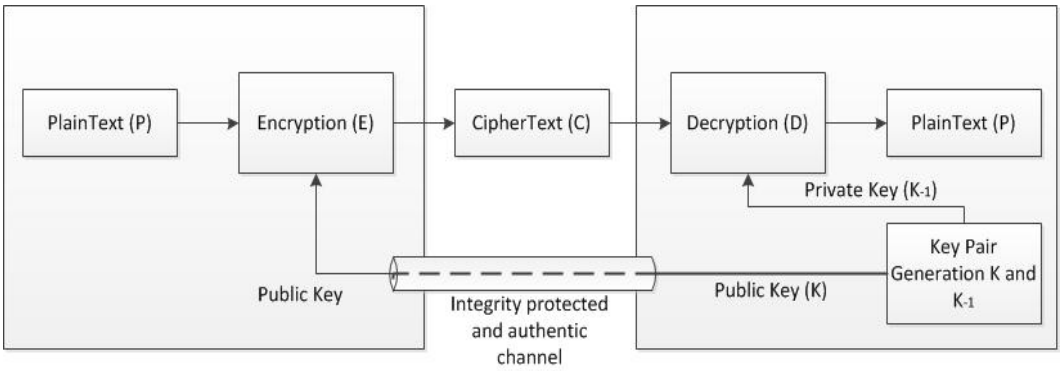

Figure 2.6: Asymmetric cryptographic cycle

Private Key: "A key that is used with an asymmetric cryptographic algorithm and whose possession is restricted (usually to only one entity)" (I.-T. X. I. 10181-1, 1996).

Public Key: "A key that is used with an asymmetric cryptographic algorithm and that can be made publicly available" (I.-T. R. X. I. I. 10181-1, 1996).

A key pair consists of a public and the corresponding private key. The generation of a key pair can be performed by different parties:

- The key pair can be generated by the decryptor. In this case, the decryptor can publish the public key in a directory service or directly send the public key to the encryptor. This case is shown in figure 2.6. Note that the channel for transporting the public key does not have to be protected in terms of confidentiality.
- The key pair can be generated by a trusted third party. In this case, the private key must be transmitted to the decryptor via a channel that is protected in terms of confidentiality.

The private key must be protected by the decryptor. Regardless of which entity undertakes the key pair generation, the public key must be made available to the encryptor. The encryptor must be confident that the public key belongs to the decryptor. This can be achieved using digital certificates (if a trusted third party is available) or by transport through integrity-protected channels with data origin authentication enabled.

Asymmetric cryptographic algorithm: The most commonly used asymmetric encryption algorithm is the RSA algorithm, named after its inventors (Rivest, Shamir, & Adleman, 1978). The RSA algorithm is based on the difficulty of factoring large integers.

2.6 XML Security Models

XML security standards present the major processing rules for the fulfilment of security requirements. Basically, XML security standards use traditional cryptographic protocols and security standards, all combined with XML technologies. The XML security

standards include XML Encryption (Imamura et al., 2002) to provide confidentiality, XML digital signatures (Bartel, Boyer, Fox, LaMacchia, & Simon, 2002) to cover integrity, XML key management (XKMS) (Hallam-Baker & Mysore, 2005) to provide public key registration and validation, XML Access Control Mark-up Language (XACML) (GODIK & MOSES, 2002) for stating authorization rules, and security assertion markup language -SAML (OASIS, 2002) to cover authentication and attribute assertions.

An XML security model needs to support the following:

1. A robust authorization mechanism whereby it can control accessibility to content and structure.
2. Ability to reuse existing security and cryptographic technologies when needed.
3. The ability to enforce security policies efficiently without the need to look up the underlying document.
4. Schema information, characterizing exactly those elements accessible to each type of user.

2.6.1 XML Signature

XML Signature was first introduced by (Bartel et al., 2002), XML Signature defines a standard format to represent digital signatures in XML, it provides a method for efficiently employing digital signatures to XML resources. However, XML Signature may also be utilized to sign binary resources like videos, images, and sound files. Several resources within XML can be covered by one signature, whether it is a whole document, part of a document, or a binary document.

W3C has established related specifications that are required when the actual XML Signature is deployed. These specifications are:

1. Exclusive XML Canonicalization Version 1.0, published by W3C (JOHN BOYER, EASTLAKE, & REAGLE, 2002).
2. Canonical XML Version 1.0, published by W3C Recommendation (J. Boyer, 2001).
3. XML Signature XPath Filter 2.0, published by W3C Recommendations (J. Boyer, Hughes, & Reagle, 2003).

The above specifications led W3C to publish a second edition (Bartel, Boyer, Fox, LaMacchia, & Simon, 2008). Figure 2.7 represents the XML Signature element's basic structure.

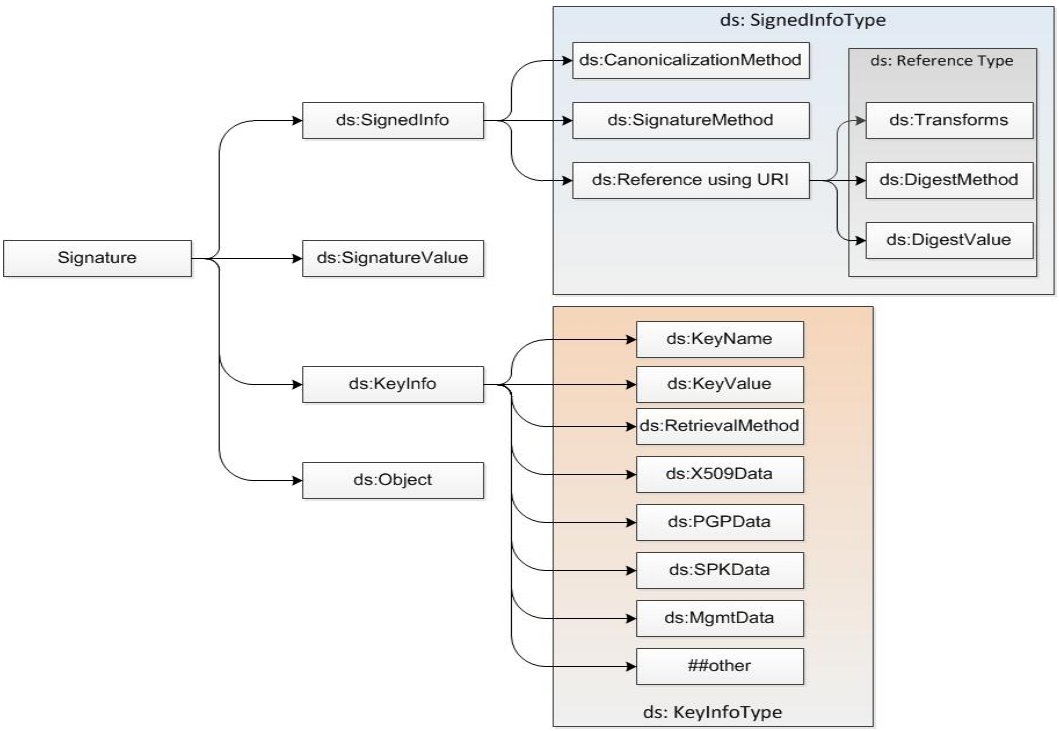

Figure 2.7: XML Signature structure

As illustrated in figure 2.7, the structure of XML Signature starts from the <*ds:Signature*> element at the top of the document. The root element <*ds:Signature*> contains four main sub-elements, which are: <*ds:SignedInfo*>, <*ds:SignatureValue*>, <*ds:KeyInfo*>, and <*ds:Object*>. The element <*ds:SignedInfo*> includes references to the applied algorithms used in XML Signature generation, the hash value, and the target

in the XML data (Weerasinghe, Elmufti, Rajarajan, & Rakocevic, 2006). Signature results are stored in the element <*ds:SignatureValue*>; element <*ds:KeyInfo*> contains the public key information that is used when the XML Signature is verified. The element <*ds:Reference*> within the element <*ds:SignedInfo*> is connected with each resource; A Uniform Resource Identifier (URI) identifies all resources. A digest of the referenced resource is included in the <*ds:Reference*> element. The <*ds:SignedInfo*> element can contain multiple <*ds:Reference*> elements. The element <*ds:SignedInfo*> contains references to the resources being signed. Therefore, an XML signature might be enveloping, enveloped, or detached taking into consideration each referenced resource. Figure 2.8 illustrates enveloped, detached, and enveloping signatures.

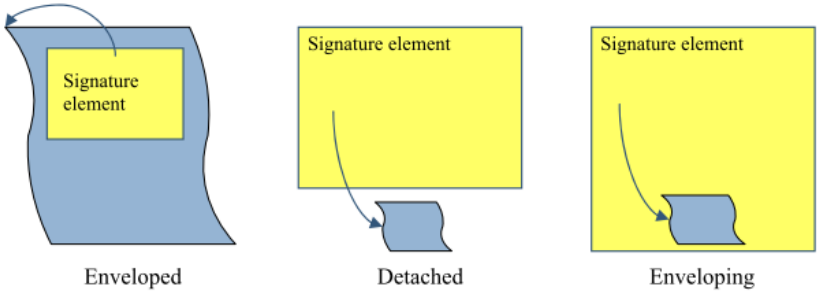

Figure 2.8: Enveloped, detached, and enveloping signatures

If the Signature element is within the referenced XML resource, then we call this signature an enveloped signature. If the signature references a resource that is separate from the Signature element, then we call this signature a detached signature. Finally if the signature references a resource that is contained within the Signature element, then we call this signature an enveloping signature. When the signature is enveloping, an instance of the object element is used to contain the resource.

As the Signature element of an enveloped signature is actually located within the XML document being signed, an enveloped signature transform is defined. This transform removes the entire Signature element from the digest calculation, so that the Signature

element is not included in the digest of the XML resource being signed. Otherwise it would not be possible to calculate the correct digest, considering that the resource (from which the digest is to be calculated) would be subject to change when adding the digest to the Signature element. XML Encryption also uses <*ds:KeyInfo*> element, this element is defined by XML Signature. The <*ds:KeyInfo*> element is extended by XML Encryption with an <*EncryptedKey*> element, which might support the transport for a symmetric/secret key. The <KeyInfo> element is used by the XML Key Management Specification as well.

2.6.2 XML Encryption

XML Encryption was first released by W3C as a proposed recommendation (Eastlake & Reagle, 2002; Imamura et al., 2002), providing encryption for different sizes of units. The units may be a whole XML document, an element within a document, XML element content, or an attribute.

XML Encryption is an encryption technology that is optimized for XML data. A format is provided for using the XML processing rules in encryption and decryption processes. XML Encryption can be performed partially, which encrypts selected tags within the XML document, or multiple times, which enables the data to be encrypted multiple times. XML can be used to facilitate resolving XML data eavesdropping.

Generally, an XML element containing encrypted XML information can act as a container for the encrypted data, keys, or both. XML Encryption is able to encrypt the whole XML document and is also capable of encrypting parts of the XML document (Geuer-Pollmann, 2002). The inclusion of encrypted content can be as a reference via the transform machine or can be included in the container. Key management is offered by XML Encryption to facilitate the symmetric wrapping of the private keys being used, private key transportation, and key agreements using a Diffie-Hellman key (Diffie &

Hellman, 1976) exchange. By using the XML Encryption standard, we gain a number of benefits, including:

- An XML element can act as a container for encrypted data, as a container for encrypted key material, or as a container for both. However, in order to act like a container, the XML element should contain XML Encryption information.
- User data can be encrypted by using XML Encryption, like:
  - Full XML documents (Geuer-Pollmann, 2002)
  - Single elements inside an XML document
  - The content of an element inside an XML document
  - Arbitrary binary content outside of an XML document
- The ability to allow direct inclusion of the encrypted content in the container.
- The ability to de-reference the encrypted content via the URI / transforms mechanism.
- XML Encryption offers key management facilities for:
  - The symmetric wrapping of secret keys.
  - Key transport of secret keys.
  - Key agreement using a Diffie-Hellman key.

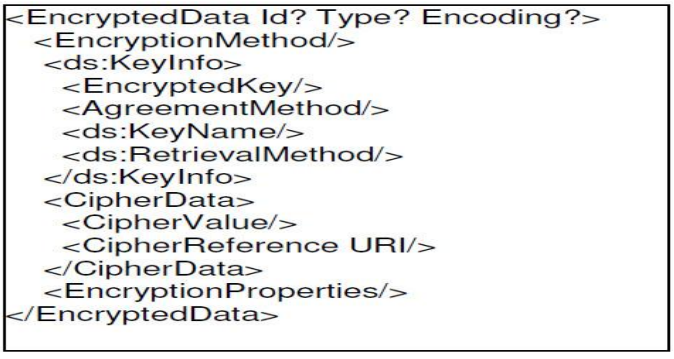

Figure 2.9: XML Encryption structure

Figure 2.9 illustrates the structure of XML Encryption provided by W3C. Data objects are encapsulated within a defined encryption element called <*EncryptedData*>. This

element contains essential sub-elements that describe how the data is encrypted; the first sub-element is <*EncryptionMethod*>, which determines which encryption algorithm is used within the XML message. The second sub-element is <*EncryptedKey*>, which is used to transport encryption keys between the sender and receiver; it can also be used individually in a separate XML message. <*KeyInfo*> is the third sub-element and is used to specify the associated keying material. Another major element is <*CipherData*>, a mandatory element that provides the encrypted data. It must contain the encrypted octet sequence of the base64 encoded text of the <*CipherValue*> element. Another way is by providing a reference to an external location that contains an encrypted octet sequence location via another element called <*CipherReference*>. Figure 2.10 and figure 2.11 represent both the <*EncryptedData*> element and the <*EncryptedKey*> element with all their sub-elements.

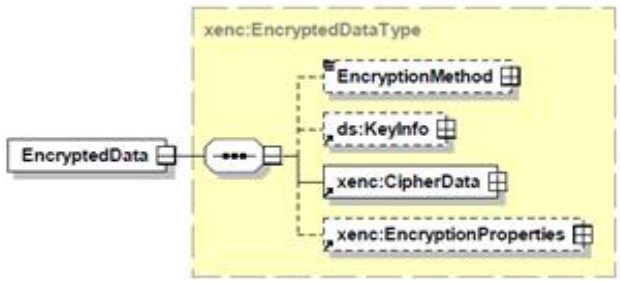

Figure 2.10: <*EncryptedData*> element and components

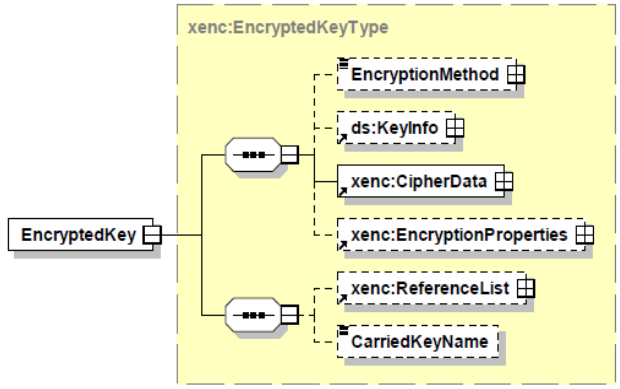

Figure 2.11: <*EncryptedKey*> element and components

The W3C XML Encryption Recommendation allows two different granularity levels: encryption of full sub-trees whereby a single element and all its descendants are encrypted, and encryption of sequences of sub-trees whereby a sub-tree can be a single node or a mixed sequence (comments, elements, text, and processing instructions).

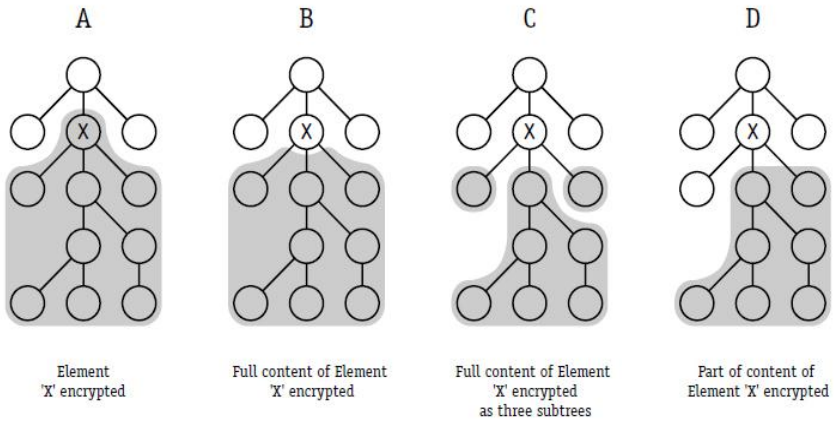

Figure 2.12: W3C Encryption possibilities (modes)

- Drawing A in figure 2.12 presents the encryption of a sub-tree rooted by the element ‘X’. The element and all its descendants are encrypted into a single <EncryptedData> element.
- Drawing B in figure 2.12 presents the encryption of the content of element ‘X’. All children of the element and their respective descendants are encrypted into a single <EncryptedData> element.
- Drawing C in figure 2.12 presents sub-tree encryption applied three times to each child of element ‘X’. Each sub-tree rooted by a child node of element ‘X’ is encrypted into a separate <EncryptedData> element.
- Drawing D in figure 2.12 presents a way to use content encryption: two subsequent sub-trees are grouped together and are encrypted together.

The decryption in drawings A and C leads to single elements. The octets resulting after the decryption in drawings B and D are not directly parseable but must be wrapped in a start tag / end tag combination.

2.6.2.1 Encryption for Multiple Recipients

1- Encrypting the same content: There are different ways to encrypt any resource intended for multiple recipients. The basic case is where all recipients have the privilege to see the same portion of the document, which means the content is encrypted only once, whereas the content encryption key is encrypted multiple times (once for each recipient). The document should include a single <EncryptedData> element for the encrypted content and an <EncryptedKey> element for each recipient, which includes the content encryption key encrypted under the recipient's key.

2- Super Encryption: When recipients are allowed to see different portions of a document, then there is a way to encrypt content for multiple recipients. Figure 2.13 illustrates the process of encrypting encrypted content multiple times.

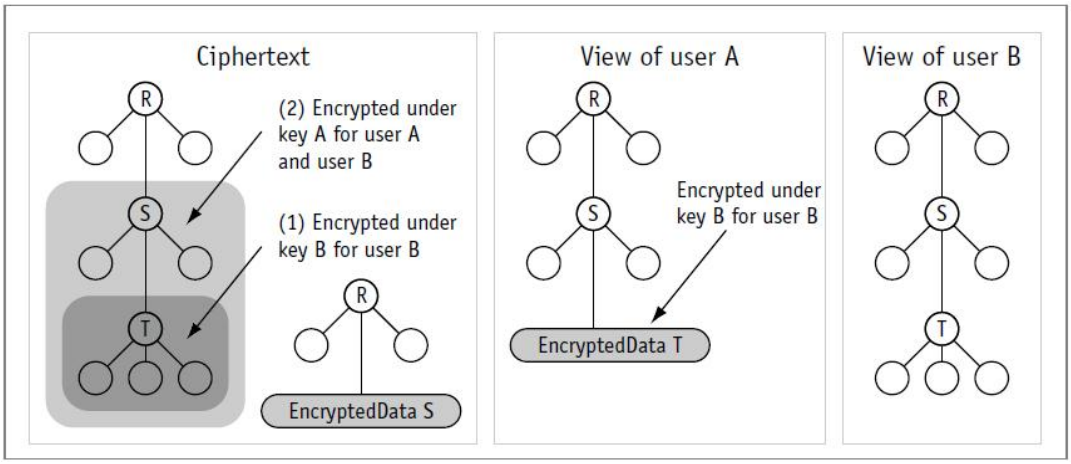

Figure 2.13: Encrypting the encrypted content for multiple recipients (super-encryption)

The process of encrypting parts of an XML tree leads to the substitution of the existing plaintext structure with the appropriate XML Encryption element <*EncryptedData*>. Super-encryption applies when the <*EncryptedData*> element or its ancestors are encrypted.

As illustrated in figure 2.13, the element 'T' sub-tree is encrypted under a key B for a recipient B. After the first step, the element 'S' sub-tree is encrypted under key A for

both recipients A and B. Keys A and B are processed by recipient B. Key A is processed only by recipient A. After the two encryption steps, the main document contains the ‘R’ element and two unencrypted notes along with the <*EncryptedData*> element, which has the encrypted element ‘S’ and its descendants. Both recipients A and B can decrypt the outer <*EncryptedData*> element because they have key A. The decrypted element ‘S’ contains the inner <*EncryptedData*> element.

The inner <*EncryptedData*> element can only by decrypted by recipient B because recipient B is the only one who possesses key B. There is a part in the document where recipient A is aware that he is not able to decrypt it. Recipient A can make an estimation of how large the plaintext (undecrypted) portion is. This estimation is based on the number of octets of the undecryptable ciphertext. Recipient B has both content decryption keys A and B and performs the decryption in two stages: decrypting the <*EncryptedData*> element containing ‘T’ is performed after decrypting the <*EncryptedData*> element that contains the ‘S’ plaintext.

After performing the decryption process, the document is decrypted in full and is available to recipient B. Recipient B acknowledges that super-encryption of the innermost <*EncryptedData*> is done in order to prevent other users accessing the inner information.

2.6.2.2 Serialization of XML for XML Encryption

Usually, symmetric encryption algorithms such as DES and AES are used for encrypting large amounts of data. Symmetric encryption algorithms transform a plaintext octet string into a ciphertext octet string and vice versa. Due to the tree-structured nature of XML, it must be converted into an octet string prior the encryption

process, and then converted back from an octet string into a tree-based structure after the decryption process.

In order to encrypt portions of a given XML document, the application selects balanced portions of XML and serializes them into a UTF-8 encoded octet sequence.

Namespace nodes and associated attributes in the XML namespace need to be taken care of: moving encrypted data into a different context can lead to inconsistent results after the decryption process. Such issues happen if the decrypted plaintext uses namespace prefixes without defining them.

2.6.2.3 Example of XML Encryption

Figure 2.14 illustrates a sample XML message fetched from a real production environment. Considering the plaintext shown in figure 2.14, this represents a financial transaction containing public information about the transaction (payee name and branch code) and sensitive information (payee account).

```
<?xml version="1.0"?>
<Transfers>
  - <Transaction xmlns="http://example.org/paymentv2" ImportanceLevel="001">
        <TransactionAmount>250</TransactionAmount>
        <From_Account>321456987456321457</From_Account>
        <Transaction_Currency Code="USD">001</Transaction_Currency>
        <Account_Type Code="001">Personal</Account_Type>
        <Payee_Name>Faisal Ammari</Payee_Name>
        <Branch_Code>Amman</Branch_Code>
    </Transaction>
</Transfers>
```

Figure 2.14: Sample XML financial message

Parts of the XML message shown in figure 2.14 will be encrypted. Figure 2.15 illustrates the sample XML message after deploying XML encryption on the selected tag <From_Account>, which denotes the full account number of the payee.

Figure 2.15: Sample XML financial message after partial W3C encryption

The <*Transaction*> element was substituted by an <*EncryptedData*> element of type element. The <*EncryptedData*> element contains the <*CipherData*> element, which uses a <*CipherValue*> element to save the encrypted <*Transaction*> element and all its descendants.

2.6.2.4 Example Implementations of XML Encryption

One of the most well-known implementations of XML encryption is XEnc (Imamura et al., 2002). XEnc is a stream-based prototype implementation that uses the Xerces Native Interface (XNI) of Xerces2. The implementation using XNI API achieves a reduction in processing time of 0.27%–26% for XML documents encryption that have sizes larger than 2KB and 34-88% reduction in decryption of XML documents of any size. Despite the reduced processing time, the issue has been raised as to whether XNI SAX API capable of parsing decrypted data efficiently.

XEnc uses DOM in many if not most of the implmentations, rather than using SAX API. preferring DOM than SAX as DOM because that DOM has the ability to parse decrypted data in an efficient way. However, Implementing DOM cost more time and space compared to SAX API due to XML document being parsed in memory.

2.6.2.5 Issues Regarding Attribute Values

XEnc uses element-wise encryption as a security mechanism, which secures both elements and content. Despite the flexible nature of XEnc. However, it is impossible to handle attribute encryption due to the XML Encryption Syntax and Processing rules. There are two solutions proposed by Simon (Ed, 2000), who first uncovered this issue. It is assumed that the alt attribute of the video element in figure 2.16 will be encrypted.

```
<video alt="C++ tutorial in 10 days" src="tutorial1.mpeg"/>
```

Figure 2.16: The alt attribute (to be encrypted) of a video element

The first solution was proposed to replace the targeted attribute that needs encryption with the <*EncryptedDataManifest*> attribute and adding other encryption details within the element. The output of this solution is illustrated in figure 2.17.

```
<video src="tutorial1.mpeg" enc:EncryptedDataManifest="./EncryptedDataManifest"
xmlns:enc="http://www.w3.org/xml/encryption/...">
    <EncryptedDataManifest xmlns="http://www.w3.org/xml/encryption/...">
    <EncryptedData Type="video/mpeg Name="tutorial1.mpeg">
        <CipherText URI="secret.enc"/>
    </EncryptedData>
    <EncryptedData Type="AttributeValue" Name="alt">
        <CipherText>AbCd...WxYz</CipherText>
    </EncryptedData>
    </EncryptedDataManifest>
</video>
```

Figure 2.17: Output of encrypting the alt attribute of the video element

The second solution was proposed to transform the attribute into elements by using XSLT to be used for encryption. However, this proposed solution is inefficient because the decrypted needs to be transformed back into attributes. Transformation is needed to validate the document against its XML Schema (if there is one).

2.6.3 XML Key Management

XKMS stands for XML Key Management Specification. XKMS Version 1.0 was first submitted to the W3C in 2001 (Phillip M & Ford, 2001). XKMS Version 2.0 was proposed and published in 2005 by Hallam-Baker (Hallam-Baker & Mysore, 2005). The main objective of XKMS is to provide a way to implement a Public Key Infrastructure (PKI) in web services and applications (King, 2003). XKMS has been developed to facilitate PKI handling, providing a simplified interface through which the application can pass. XKMS simplifies PKI handling by moving the complexity of dealing with the PKI from the application to the XKMS service itself. The application is thus protected from primary complexities (O'Neill, 2003).

In both cases, X-KRSS provides mechanisms for authenticating clients. X-KISS defines two main services: locate service and validate service. <KeyInfo> element provides the data format that is needed for communicating key information; this element is defined by XML Encryption. Thus, it will facilitate the utilization of XKMS together with XML Encryption and XML Signature.

XKMS consist of two parts. The first part is called X-KRSS, which stands for XML Key Registration Service Specification. The second part is called X-KISS, which stands for XML Key Information Service Specification. X-KRSS defines the services for the processes of registering, revoking, recovering, and reissuing keys. The client or provided service can perform the process of new public keys registration. If the client generates the key pair, then the client is required to provide or present the authenticity of their owning the private key to be able to register for public key. Regardless of whether the client or the provided service generates the key pair, KRSS provides mechanisms to authenticate clients. There are two services defined by X-KISS: locate service and validate service. The locate service allows a client to fetch information

about a public key or it allow a client to fetch a public key. The validate service allows a client to fetch information about a public key or it allow a client to fetch a public key but it confirms that the information returned matches specific validation rules. The <KeyInfo> element defined by XML Encryption is used to provide the data format that is used to communicate key information.

Figure 2.18 illustrates how XKMS operates in steps. In figure 2.18, sender B target is to submit the encrypted document to sender A using the public key in possession. Though, sender B does not have sender A's public key. Although sender A has the public key registered using XKMS service, within the sender A's domain there is no trust relationship established between sender B and the XKMS service. This can be resolved by sender B contacting the validate service within sender B's domain, requesting a public key for sender A to be used in the encryption process. This request might be forwarded by the validate service to the locate service within sender A's domain. Validate service will validate the response within its own domain; validation process is performed before the sender B has the response returned.

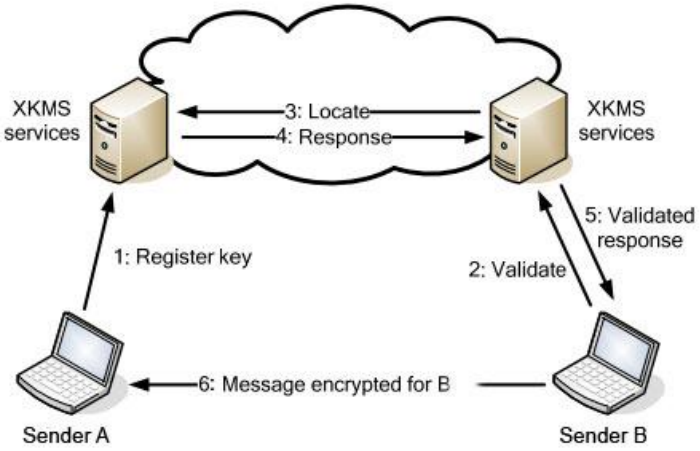

Figure 2.18: Obtaining a validated public key by sender B for sender A

2.6.4 Security Assertion Markup Language (SAML)

SAML stands for Security Assertion Markup Language (Cantor, Kemp, Philpott, & Maler, 2005). SAML defines the representation of security assertions within XML documents. An assertion process is a set of pre-defined statements, created by an assertion authority, which a relying party may trust. Figure 2.19 represents the assertion process, whereby the required assertion is identified by the issuer element. There are three statement types defined by SAML: authorization, authentication, and attribute statements. The same abstract type derives the three statement types, from which any additional statement types may be derived as well. Any number of statements can be included in a SAML assertion. In cases where the assertion has all three assertion types, it is necessary to indicate the subject type to which assertion type can be applied, in order to utilize the subject element. Subjects' confirmation methods can be specified by the subject element. Such methods can be used to ensure that the message origin is exactly from the subject identified in the assertion. There are many methods for subject confirmation (Hughes et al., 2005). Usually, the message is signed by the subject and by using a private key associated with the assertion. Signing the message can be performed by other applications' specific procedures.

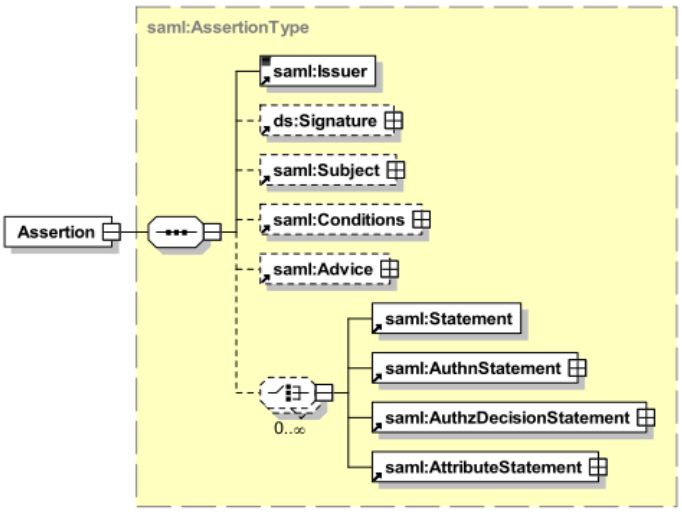

Figure 2.19: SAML assertion elements

2.6.5 XML Extensible Access Control Markup Language (XACML)

XACML stands for Extensible Access Control Markup Language and is an open standard XML-based language designed to explain the security policies and access privileges to data and information for digital rights management (DRM), web services, and enterprise security applications. XACML was first introduced by OASIS (Organization for the Advancement of Structured Information Standards) in 2003 (Godik & Moses, 2003). The main objective of XACML is to set a basic standard for access control through XML language.

XACML can work in conjunction with SAML, whereby a rule engine with policies expressed in XACML can compare such information with established criteria to ascertain user rights.

Figure 2.20 illustrates the basic components of XACML. As seen in figure 2.20, the policy enforcement point and the policy decision point can be shared with SAML.

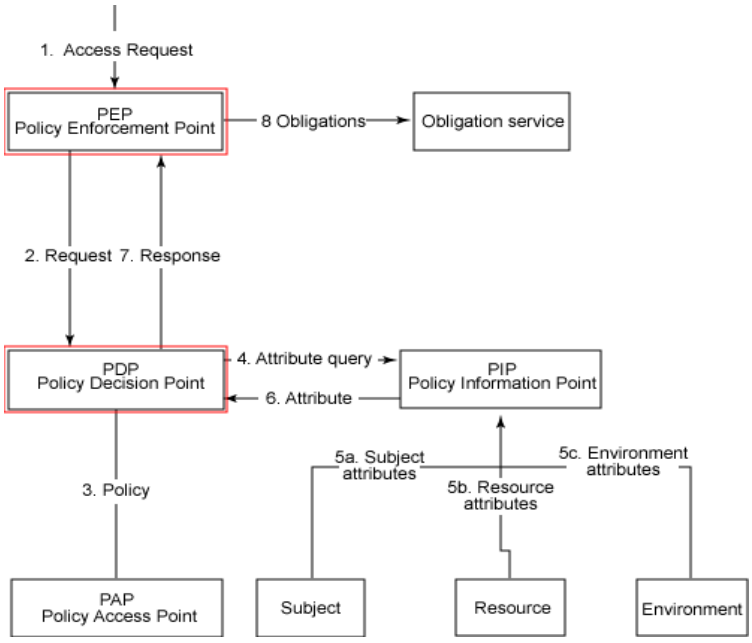

Figure 2.20: XACML components

An access request arrives for authorization at the Policy Enforcement Point (PEP). An XACML request is created by the PEP and submitted to the Policy Decision Point

(PDP), which then evaluates all incoming requests and sends it back with responses (either accepted or denied).

A decision is made by the PDP after evaluating the relevant policies and the rules within them. Not all policies are evaluated; only relevant policies are picked up for the evaluation process, which depends on the policy target.

The Policy Access Point (PAP) is used by the PDP to fetch all policies, write policies and policy sets, and to ensure the policies are available to the PDP. In order to retrieve the attribute values associated with the subject, resources, or the environment, the PDP may invoke the Policy Information Point (PIP). The PEP fulfils the needed requirements once the authorization decision arrived at by the PDP either permits or denies access.

2.7 Text Categorization

Text categorization is the process of assigning text documents to a predefined set of categories/classes. There are two main phases involved in the categorization process: the training phase and the fuzzification phase. In the training phase, sets of documents belonging to each category are used to create representations of the categories. The classification phase compares the representations resulting from the training phase with a new document in order to assign a new document to one or more category. Many algorithms that can be used to perform the process of text categorization, including k-nearest neighbour classification (kNN) (Guo, Wang, Bell, Bi, & Greer, 2006), support vector machines (SVMs) (Brank, Grobelnik, Milic-Frayling, & Mladenic, 2003), neural networks (Ng, Goh, & Low, 1997), linear least squares fit mapping (LLSF) (Yang & Chute, 1993), the vector space method (Gauch, Madrid, Induri, Ravindran, & Chadlavada, 2004), and naïve Bayes classification (NB) (McCallum & Nigam, 1998).

(Yang & Liu, 1999) conducted a performance comparison of all these classification systems and found that SVMs and kNN significantly outperform all other classifiers.

2.8 Fuzzy Logic Model

Fuzzy logic (FL) was first introduced by (L.A. Zadeh, 1965). Fuzzy logic is based on multi-value logic, which allows intermediate values to be defined between conventional evaluations in the form of Yes/No, True/False, etc. Different concepts like “very short” or “rather slow” can be formulated in mathematical notations and processed by computers, in order to apply a more human-like way of thinking (L. A. Zadeh, 1984).

FL provides a powerful method to reach a certain conclusion that is based on vague, uncertain, or noisy information. In order to make faster decisions, FL provides a mechanism to control problems. FL provides the ability to integrate rule-based approach (consist of IF X and Y THEN Z rules) to solving control problems.

The FL approach provides essential information to assist financial decision makers in effectively measuring and identifying sensitive information, essential parts and important figures within financial transactions, more so than the existing qualitative approaches. This is because, using FL, the degree of importance within each financial transaction is quantified based on a combination of financial historical data and input by experts.

Over the years, FL has been used to integrate expert input into computer models for a large scope of applications in different categories. The main advantage of the fuzzy methodology is that it enables the processing of ambiguously defined variables and variables whose relationships cannot be defined by any mathematical relationships. In order to define those variables along their relationships, fuzzy logic can incorporate expert human judgement for that purpose.

2.8.1 Fuzzy Sets and Crisp Sets

The fuzzy subset is the basic concept of fuzzy systems. In mathematics, we call it a crisp set. Figure 2.21 illustrates the characteristic function of a crisp set.

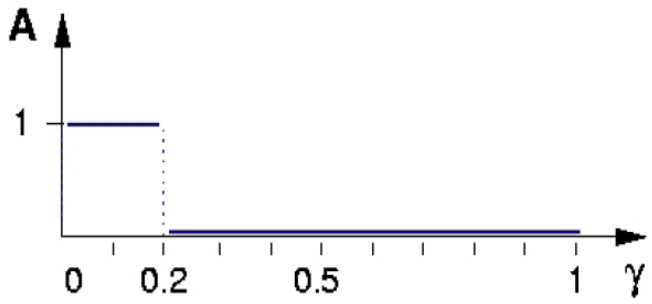

Figure 2.21: Characteristic function of a crisp set

In figure 2.21, the elements that have been assigned the number 1 can be interpreted as the elements that are in set A, whereas the elements that have been assigned the number 0 are the elements that are not in set A. This concept lacks flexibility for some applications. The upper range is difficult to define. So, the upper range is set to 0.2. Thus, we get B as a crisp interval defined as following: (B= [0, 0, 2]). However, this means that a value of 0.21 is not considered low while a value of 0.20 is considered low. To construct the set B in a more natural way is by relaxing the strict separation between low and not low. This can be achieved by allowing more flexible rules like "fairly low" rather than allowing only the crisp decisions. A fuzzy set will enable us to define such a notion.

The intention is to utilize fuzzy sets to be able to make computers more 'intelligent'. And so, the idea above needs to be coded more formally. A straight method to generalize this concept would be to enable more values between 0 and 1. Actually, infinitely many choices could be permitted between the bounds 1 and 0, specifically the unit interval I = [0, 1]. A gradual membership is reflected by all other values to establish

B. This really is proven in figure 2.22. The membership function is a graphical representation of the magnitude of contribution of every input signal. It associates a weighting with all of the inputs which are processed, identifies functional overlap between inputs, and ultimately determines an output result. The rules utilize the input membership values as weighting factors to ascertain their influence in the fuzzy output sets of the final output conclusion.

The membership function, functioning in this instance in the fuzzy set of interferometric coherence g, returns a value between 0.0 and 1.0. It's significant to indicate the distinction between probability and fuzzy logic. Both operate over exactly the same numeric range and have similar values: 0.0 represents non-membership (or False) and 1.0 represents full membership (or True). However, there's a distinction to be made between the two statements. The probabilistic approach yields the natural language statement, "There is a 50% likelihood that g is low", as the fuzzy language corresponds to "g's degree of membership within the set of low interferometric coherence is 0.50." The semantic difference is critical: the first view supposes that g is or isn't low; the chance to know which set it is in is only 50%. Fuzzy terminology assumes that g is "more or less" low or, corresponds to the value of 0.50.

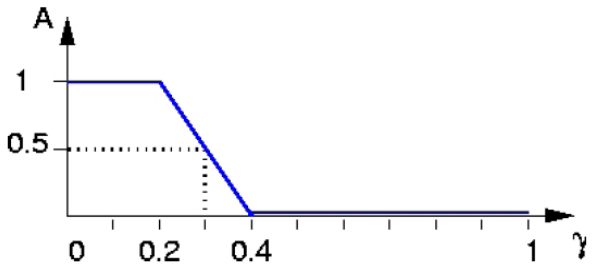

Figure 2.22: Characteristic function of a fuzzy set

2.8.2 Fuzzy Inference Process

FL can allow the use of degrees of truth in order to calculate results. FL techniques allow one to represent concepts that could be considered to be in more than one

category. which means the representation of overlapping and partial membership in sets or categories is allowed (Bridges & Vaughn, 2001). There are four main steps involved in the FL inference process (Cox, 2001b), which are:

- Step 1 (Fuzzification): Taking the crisp input X and input Y, the process determines the degree to which these inputs belong to and where they fit in the fuzzy set.
- Step 2 (Rule Evaluation): Taking the fuzzy inputs, the qualified fuzzy rules are applied. Fuzzy operators (AND / OR) are used in case of any uncertainty to get a single value. The outcome value is called a "Truth Value", which will be applied to the membership function for rule evaluation.
- Step 3 (Aggregation of the Rule Outputs): The outputs of all the rules are unified. Scaled rules are combined into a single fuzzy set for each variable.
- Step 4 (Transforming the Fuzzy Output into a Crisp Output): The output should have a clear, crisp value and it will be assigned to each tag classified.

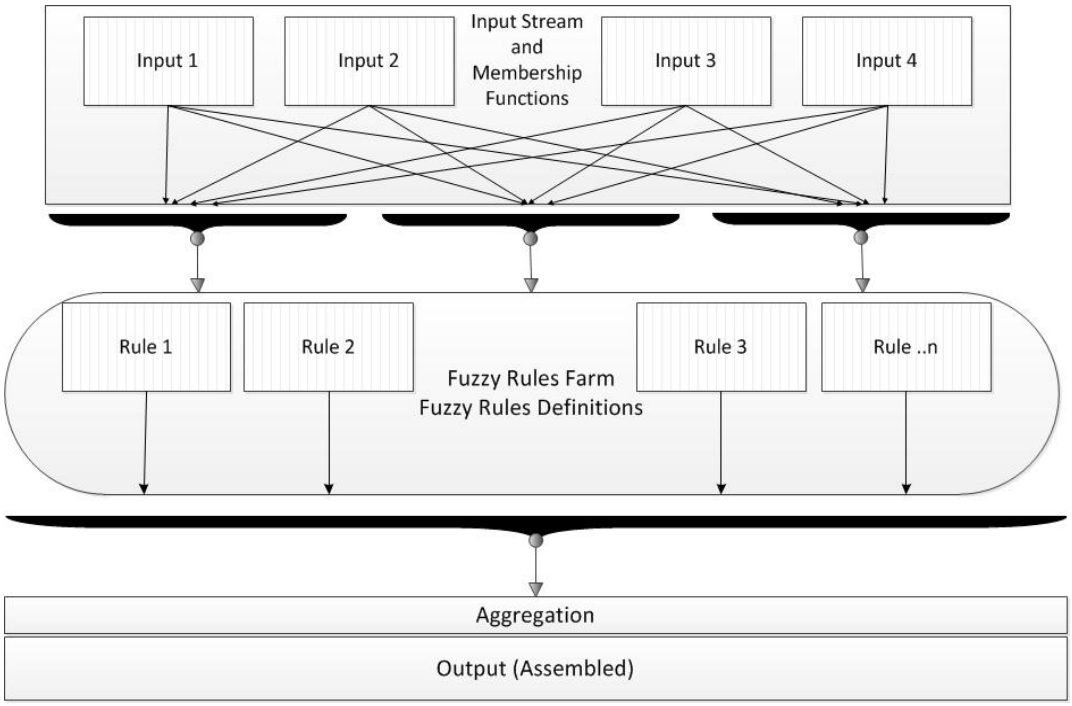

Figure 2.23: Fuzzy inference process

Figure 2.23 illustrates the FL inference process, showing the four major steps required to perform the FL inference lifecycle.

Fuzzy logic is needed because of the ability that FL has to accept vaguely defined data. It has the ability to model non-linear functions of arbitrary complexity and can build on the experience of experts.

2.9 Chapter Summary

This chapter introduces the XML model and its basic components and the schema languages involved. This chapter also introduces the main XML security techniques that are utilized in XML security specifications. The XML security specifications published by W3C and OASIS are the core of XML security technology, such as XML Encryption. XML Encryption makes XML data confidential and it ensures this using XML Encryption technology.

Text categorization has been discussed as well to describe the classification methods for the text documents and their associated algorithms to categorize data into relevant categories properly.

Finally, fuzzy logic has been introduced to explain the methodology we are using in this research to identify how we will classify XML content and find the importance level for each tag included within the XML document.

Chapter 3

Literature Review

3.1 Introduction

This chapter analyses the most important XML encryption models. Models are the key factor inspiring and supporting this research. Classifying XML messages using fuzzy logic (FL) is the other key part of this research and is also discussed and analysed in this chapter. We explore existing XML encryption models and a list of the related works.

3.2 XML Encryption Models

Companies and financial institutions adopted XML as a standard in their data communication and exchanges among different platforms due to its independency, flexibility, and ability to enable custom structures and creations. XML documents are exchanged via variant communication mediums, which in some cases have weak security measurements; they may use unsecured channels or may even be hacked by an unknown party. XML documents are well known to be verbose, which make them readable by machines and humans as well. Any plaintext editor should be enough to read, modify, or change the contents to cause real damage.

Therefore, some XML encryption models have been proposed and published to ensure the confidentiality of transferred XML messages and to make sure they reach their destinations without any issues.

(W3C, 2001) first published the XML Encryption Standard (XEnc) by describing the syntax needed to represent encrypted XML data and the process of encrypting and decrypting XML data. Data confidentiality is achieved by the XML encryption by hiding sensitive information so that it can only be understood by targeted recipients. The XEnc structure syntax is defined by using XML Schema.

Element-wise encryption is the security provided by XEnc, where elements and content are integrated with each other. Due to the flexible and extensible nature, the XEnc standard can be used and processed by XML tools. However, it is impossible to expand their capability handling attribute encryption due to the current XML Encryption Syntax and Processing rules.

(Maruyama & Imamura, 2000), (E & B, 2000), and (Takeshi & Hiroshi, 2000) have worked on XML element-wise encryption, which supports W3C in delivering a candidate specification for XML encryption (Imamura et al., 2002). The candidate specification specifies a process for encrypting data and representing the result of the encryption process in XML. Data might be in different format, for example it could be arbitrary data, XML elements, or XML element content.

The encrypted data element is called <*EncryptedData*>. This element identifies the format of the encrypted data; the identification does not include user's ability to identify how the XML document is encrypted. It is impossible to handle attribute encryption due to the <*EncryptedData*> element's syntax.

(Ed, 2001) explained the attribute encryption as following: Elements and attributes are identified by XML for information structuring and applications that uses attributes in a frequent basis. Author urged the support of attributes encryption unless it is impossible to support.

(Steve, 2001) explained the difficulty of redesigning a legacy application and XML vocabularies in cases of existing attributes containing data. Such redesigning can be onerous when there are pre-existing XML data and applications. The main goal is to find a simple yet effective way of handling the encryption of attribute data.

(D. Fallside, 2001) presented some negative sides of attribute encryption that might trigger issues. The first possible issue is that the encrypted XML document cannot be validated against the original XML schema. This will result amending the original schema so that it can identify particular encrypted elements. Briefly, assuming an XML document (X) with its schema (S) defining both structure of the document and content. Next, the encrypted document (Xs) will not follow (S), because it is impossible to present XML encryption without schema changes during the encryption process, as described in (Blair, 2001) and (Ed, 2001).

Without attribute encryption in XML, sensitive data cannot be stored securely in the attributes of XML documents. Also, we have to tell the users to redesign their legacy XML documents if they wish to apply XML encryption to them. We consider that the only acceptable reason for not including attribute encryption in XML is if it is impractical or impossible to do so. After much discussion about the requirements, complexities, and alternatives of attribute encryption, the working group decided to proceed under the requirement of element encryption while remaining open to further comment, experimentation, and specification of attribute-encryption proposals or

alternatives that satisfy the requirement to encrypt sensitive attribute values (Joseph, 2001). (Ed, 2001) also mentioned the possibility of leaving out attribute encryption until version 2.0 of XML encryption.

(Geuer-Pollmann, 2002) proposed an encryption approach called pool encryption: the approach has the ability to remove sensitive information from the resulting file. Each XML document is parsed into a DOM tree for encryption purposes. DOM tree nodes are labelled, the corresponding node has the position information attached to it. Each node is individually encrypted whereby it has an encryption key specified for each node. These nodes are removed later from their unique position located in the source document and placed into a pool of encrypted notes.

The pool of nodes can be stored either in different documents or in the source document; the choice will depend on the pre-defined security requirements. A unique node key is assigned to each node. The node keys are grouped into a pool of node keys to ease nodes management. The sender identifies the decryption competences of the different users by assuring that the recipient has the pool of node keys. The more keys the pool receives, the more nodes it can decrypt. The recipient will only be able to view parts within encrypted document taking into consideration the corresponding pool of node keys. This is because the pool is encrypted with the recipient's key before it is submitted to the recipient. Any node that does not match keys will be hidden and the recipient will not be able to view it. The individual nodes which have the bundled original position information uses this information during decryption process in order to restore any specific node back to its original position within the XML document. The pool encryption method is able to remove confidential information from the document; this method is also able to hide both the size and the existence of encrypted content. By performing this method, it prevents information leakage to unauthorized users to access

the document. However, there are number of disadvantages related to this method due to the change in the size of the encrypted document to prevent any traffic analysis attack. Disadvantages defined as following:

- The encrypted document size is changed due to the pool of node keys.
- Increase in encrypted document size due to added position information.
- For each node, the encryption process and decryption process are executed separately.
- Each node has to have a unique node key generated.
- The original position information has to be attached to particular individual nodes.
- At the contrary of the ordinary encryption/decryption processes, the process of reassembling and flattening nodes requires intensive resource usage.

The above disadvantages cause high memory usage, large storage requirements, high bandwidth consumption, and high processor power consumption.

(Rosario, 2001) introduced the concept of XML access control (XAC), which is a server-side access control whereby a trusted access control processor allows security policies and procedures to be established based on policies. XAC presents a way to control the access of users to specific portions of a full XML document that is stored on a server.

XAC encrypts an XML element with the ability to exclude its descendants. This specific feature of XAC gives an advantage over XEnc because XEnc requires the encryption of a full sub-tree.

XAC pruning process refers to the authorization of a user and in the next step it removes elements labelled with "deny (-)" from the tree.

Thus, it declines the requesting user from reading sensitive information and also it prevents the user from gaining information about any existing sensitive content. In brief, only elements with "permit (+)" labels remains in the resulting file. The pruning process is executed online at the same time the client issues a query (similar to SQL query). It is similar to XEnc but without encryption after pruning process. However, a form of additional security is needed (like SSL) to transmit the document in a secure mode to users.

However, XAC restricts the user's access to a document. Some of the techniques used: mandatory access control, rule-based access control, access control list, discretionary access control, role-based access control, and lattice-based access control.

Table 3.1 illustrates the three main XML encryption approaches with their advantages and disadvantages.

Table 3.1: XML Encryption Approaches

Approach	Advantages	Disadvantages
W3C XML Encryption Standard (2001)	• Only users that know the key can decrypt and read the message. Each recipient can only decrypt the parts of a message that are intended for them; they are unable to decrypt the rest. • *EncryptedData* is an XML element that replaces the data to be encrypted. The *EncryptedData* and the	• Unable to handle attribute encryption. • Leaves descendants visible. • Needs style sheets. • New recipients cannot be added without re-encrypting the content. • Neither of DTD nor schema definition is encrypted, both are exposed to "plaintext attack". Therefore, there is a risk of

	EncryptedKey are composed of other sub-elements, such as the encryption method, key information, and cipher value. • The entire XML message or only some parts can be encrypted. • If both the sender and the receiver have not exchanged the keys previously, the key can be sent in the message encrypted using a public key system.	information leakage.
XML Pool Encryption (Christian Geuer-Pollmann – 2002)	• Uses a secure, complete sub-tree. • Has the ability to secure attribute values. • A new recipient can be added without re-encrypting the content.	• The original position information has to be attached to particular individual nodes. • The encrypted document size is changed due to the pool of node keys. • For each node, the encryption process and decryption process are executed separately. • Each node has to have

		a unique node key generated. • Increase in encrypted document size due to added position information. • At the contrary of the ordinary encryption/decryption processes, the process of reassembling and flattening nodes requires intensive resource usage.
XML Access Control (XAC) (Ricardo Rosario – 2001)	• Involves the automation of encryption/access decisions. • Attribute encryption is possible.	• Needs additional transport security. • Needs trustworthy servers.

3.3 Fuzzy XML Modelling

Over the years, fuzzy systems have been used successfully in many fields to handle the imprecise and uncertain information that is commonly found in real-world applications, such as business and financial systems, data mining, and decision-making systems.

(L.A. Zadeh, 1965) addressed the representation of fuzzy information with fuzzy set theory. Fuzzy set theory has been successfully identified as a working technique to model imprecise and uncertain data and it has been introduced into many application areas, such as business intelligence (Petrovic, Roy, & Petrovic, 1999; R. R. Yager, 2000; Ronald R. Yager & Pasi, 2001) and database, semantic web, and information

systems (Galindo, 2008; Klir & Yuan, 1995; Lukasiewicz & Straccia, 2008; Ortega, 2008; Smets, 1996).

XML is not able to represent and process vague and unclear data. Currently, less research has been done on modelling and querying imperfect XML data. XML documents with incomplete information have been researched by (Serge Abiteboul, Segoufin, & Vianu, 2006) and probabilistic data has been researched by (Nierman & Jagadish, 2002). (Lee & Fanjiang, 2003) developed a fuzzy-object-oriented modelling technique based on the XML language to model requirement specifications and incorporated the notion of stereotypes to facilitate the modelling of imprecise requirements.

(Gaurav & Alhajj, 2006) presented an approach to integrate fuzziness with XML. They base their approach on identifying possible entities in XML that might have fuzzy values. Their approach based on analysing XML document structure to identify which parts within the document that can be handled using fuzziness. They then specified the appropriate mechanism to integrate fuzziness. Their approach focused on XML being a structured (logical and physical) and well-formed language. They were interested in the logical structure as a key issue, defining the content (data) of an XML document. Then, they identified different parts of an element that can have fuzzy data. there are five items that might have fuzzy data: 1) simple elements “text only”; 2) complex or empty elements - the attribute value "val"; 3) complex element -only the element "val" and occurrence of <subElmt> within <elmt>; 4) complex or text only elements - attribute value "val" & "text"; 5) complex and mixed elements - attribute value "val", element content "text", and the occurrence of <subElmt> within <elmt>. This means that the elements content “text”, the attribute value "val", and the <subElmt> of an <elmt> might be fuzzy entities within an XML document. this is declared from their evaluation

of including the attribute "val", fuzzy data, and element content "text" as they are similar in nature and are all equivalent, as opposed to the <subElmt> of an <elmt>. Therefore, there are two main categories of entities in an XML document that might be fuzzy: the first category is when having an element content "text", or the attribute value "val", and the sub-element <subElmt> of an element <elmt>.

However, their approach does not tackle the production of nested, fuzzy XML schema or the provision of techniques to make vague queries on a vague XML document.

(Ma & Yan, 2007) introduced a fuzzy XML data model to manage fuzzy data in XML, based on possibility distribution theory, by first identifying the multiple granularity of data fuzziness in UML and XML. A fuzzy UML data model and a fuzzy XML data model that address all types of fuzziness were developed. Further, the author developed the formal conversions from the fuzzy UML model to the fuzzy XML model, as well as the formal mapping from the fuzzy XML model to fuzzy relational databases. It is noted that the fuzzy extension of XML in the author's model only focuses on XML DTD, because it has traditionally been the most common method for describing the structure of XML instance documents. However, XML DTD lacks enough expressive power to describe highly structured data properly, and XML Schema provides a much richer set of structures, types, and constraints for describing data.

(Tseng, Khamisy, & Vu, 2005) presented an XML methodology to represent fuzzy systems for facilitating collaborations in fuzzy applications and design. DTD and XML Schema are proposed to define fuzzy systems in general. One fuzzy system can be represented in different formats understood by different applications using the concept of XSLT style sheets. As an example, they represent a given fuzzy system in XML and transform it into comprehensible formats for Matlab and FuzzyJess applications.

(Tseng et al., 2005) methodological components consist of: a) an input base that consists of a collection of inputs (linguistic variables) containing terms and membership functions; b) a membership function repository that contains all the membership functions used to describe the fuzzy system; c) an inference engine that defines all operators used to perform inferencing; d) an operator repository that contains all the operators (“And”, “Or”, “Aggregations”) used to describe the fuzzy system; e) a rule base that is a collection of fuzzy IF-THEN rules; f) defuzzification, which translates fuzzy set output values into crisp values; and g) an output base that consists of a collection of outputs (each being a linguistic variable data type).

On the lower side of their methodology, (Tseng et al., 2005) proposed fuzzy system data types that consist of: linguistic variables, linguistic terms, membership functions, operators, and rules. They then proposed a DTD as a kind of schema to describe fuzzy systems in XML; one DTD is defined per component and main data type. Fuzzy system schema can also be used to define fuzzy systems in XML; they define an individual XSD for each major component.

However, Tseng did not pay much attention to the design of reverse style sheets to transform fuzzy system descriptions in given software into an XML document that is compliant with their proposed XML schema.

(Turowski & Weng, 2002) introduced a formal syntax for the important fuzzy data types that are used to store fuzzy information. They defined appropriate DTDs as they show how fuzzy information, whose description is based on these DTDs, can be exchanged between application systems by using XML. As a result, they introduced a better approach for business applications integration using fuzzy approaches to business application systems. This allows better collaboration with application systems and development tools that use fuzzy approaches. Their approach focuses on encapsulating

fuzzy information, and any related fuzzy data that describes fuzzy information in XML tags is named according to a standardized term set. By performing encapsulation, the messages that contain fuzzy information from other application systems get a meaning and can subsequently be processed. They defined a DTD as defining constraints on the logical structures of XML documents.

(Zhang, Ma, & Yan, 2013) proposed an approach along with an automated tool called FXML2FOnto for constructing fuzzy ontologies from fuzzy XML models. They also investigated how constructive fuzzy ontologies may be useful for improving some fuzzy XML applications (i.e. reasoning in fuzzy XML models). They first proposed a definition of fuzzy XML models that includes the XML document structure's fuzzy DTDs and the XML document content's fuzzy XML documents. Based on this definition, they proposed an approach to constructing fuzzy OWL DL ontologies from fuzzy XML models.

They used two key steps to construct the fuzzy OWL DL: first, transforming the fuzzy DTD into a fuzzy ontology at the structure level; second, transforming the fuzzy XML document into a fuzzy ontology at the instance level. They gave proof of the correctness of the transformation by providing a detailed construction example. Following the proposed approach, they implemented a prototype tool called FXML2FOnto, which automatically constructs fuzzy OWL DL ontologies from fuzzy XML models.

In the final stage of their approach, they reduced reasoning in fuzzy XML models to reasoning in fuzzy OWL DL ontologies, so that by using existing fuzzy ontologies' reasoning, the reasoning of fuzzy XML models can be automatically checked. The reasoning results may provide several simple optimization steps in answering queries over a fuzzy XML document base. By using this approach, it is possible to improve some fuzzy XML business and financial applications.

(Herrera-Viedma, Peis, Morales-del-Castillo, Alonso, & Anaya, 2007) proposed an evaluation model for websites that are based on XML documents that are user centred and based on a fuzzy linguistic approach. The evaluation model consists of two components: an evaluation scheme that contains the evaluation criteria to be considered in the website quality evaluation; and a computing method of linguistic quality ratings. In the evaluation scheme phase, they analysed the information quality of websites from the information user's perspective, considering the following: a) different quality approaches to information quality; b) generating quality ratings on websites provided by evaluators; c) not including an excessive number of quality dimensions to avoid conflicting users; and d) analysing websites that store information in multiple types of documents structured in XML format (i.e. scientific articles). Based on these considerations, they defined a user-centric evaluation scheme of websites that anticipates four quality categories with the following evaluation dimensions: 1) intrinsic quality of websites; 2) contextual quality of websites; 3) representational quality of websites; and 4) accessibility quality of websites. The second component in their model is a computing method of linguistic quality rating that is used to evaluate the information quality of websites that are based on XML documents. Linguistic ratings are obtained from the linguistic evaluation judgements provided by a non-determined number of web visitors. After a visitor has used an XML document stored in a website, they are invited to complete a quality evaluation questionnaire as per the quality dimensions created in the evaluation scheme.

Ratings of the linguistic quality are obtained by performing an aggregation function of the linguistic evaluation judgements by means of the LWA and LOWA operators, which are a linguistic family of OWA operators (R. R. Yager, 1988). These operators are used to allow inclusion of the concept of a "fuzzy majority" (Herrera, Herrera-

Viedma, & Verdegay, 1996) in the computation of the rankings. The “fuzzy majority” is represented by the linguistic quantifier that is used to compute the weighting vector of the OWA operator.

However, their model has the following limitations. First, it uses little information about web users; the model is designed to compute quality ratings only. Therefore, the performance could be improved if user profiles are used in the computation process of quality ratings. Second, it is a user-dependent model, so the quality of the websites can be evaluated only if users’ perceptions can be gathered.

3.4 Chapter Summary

In this chapter, we have presented various approaches to XML encryption standards. It discussed the capabilities, functions, and mainly primary criteria of the XML Encryption Standard.

During the study of existing approaches, various issues have been addressed, such as super encryption, attribute value encryption, and other security issues. XML encryption models have been compared with one another. Implementations are considered as well for a few of the models.

W3C may have to reconsider attribute value encryption, because it's not supported by the present XML Encryption Standard. If not, confidential information should be stored inside an element, consequently deprecating the use of attributes.

We have evaluated the W3C XML Encryption Syntax and Processing rules. The security offered by XEnc is element-wise encryption, content and comprising elements. It may be utilized together with XML Signature. Its flexibility and extensibility enables XEnc to be processed and used by XML family tools. However, it is impossible to extend the capability of the current XML Encryption Syntax and Processing rules to

handle attribute encryption. We have also discussed XML access control (XAC) and made a comparison with XEnc: attribute encryption is possible in XAC. As its descendants can be excluded by the encryption of an element, XAC has greater flexibility over XEnc. This isn't permitted in XEnc as it requires the encryption of a complete sub-tree. However, XAC restricts the user's access to a document using several techniques, such as: discretionary access control, mandatory access control, role-based access control, rule-based access control, and lattice-based access control.

Pool encryption has also been reviewed as one of the encryption models that are used to encrypt XML documents. XML pool encryption has the capacity of removing sensitive content from the encrypted XML document. However, this model introduces a number of disadvantages, like the increase in the size of the encrypted document due to added position information. Such disadvantages lead to high storage, high bandwidth and memory requirements, and high processor power consumption.

Additionally, we have reviewed a number of fuzzy XML models and studies. Some of the studies tried to encapsulate fuzzy system descriptions in common elements that can be used to represent any fuzzy system. Some models proposed a DTD/XML schema to describe fuzzy systems in XML. Also, we have introduced studies presenting fuzzy XML data models that can be used to manage fuzzy data in XML, based on possibility distribution theory. Some models presented an approach to incorporating fuzziness in XML by identifying the possible entities in XML that might have fuzzy values. Potential entities that can handle fuzziness are identified by analyzing the XML document structure to incorporate fuzziness.

Chapter 4

Intelligent Fuzzy-Based Financial XML Security Model

4.1 Introduction

This chapter proposes a secure XML management model named SXMS. The model consists of two major parts. Each part has a discrete scope acting as an independent unit and forming an essential part of the whole system. Content is classified using a set of fuzzy classification techniques and encrypted using an element-wise encryption on selected parts within each XML message. This chapter also describes the combination of the model with XML message requirements and specifications.

The proposed model has been designed based on two major phases, each with a discrete scope acting as an independent unit and forming an essential part of the whole system. Phase one of the proposed model involves performing a set of fuzzy classification techniques on the targeted XML messages. The fuzzy classification process is designed mainly to decide the similarity between the different standards within the same message. Basically the main target is to classify XML content to find out which parts are essential to have a specific security standards deployed. Upon fuzzy classification, a new value is generated and assigned to an existing XML tag. The XML tag named "Importance Level" will be used as an identifier for the next phase. The “Importance

Level" tag presents a value which is used to identify the sensitivity level for the carrying tag and corresponding nodes. Phase two involves applying element-wise encryption to different parts within each XML message. Encryption could be for the whole message or elements of an XML message. The "Importance Level" value assigned in phase one is also used to decide which type of encryption and key size is to be deployed. Element-wise encryption is based on W3C's recommendation (Maruyama & Imamura, 2000).

4.2 Proposed Model for Securing XML Financial Documents

This work presents a novel approach to securing XML financial messages by using a combination of fuzzy classification techniques and element-wise encryption. Main reason behind choosing FL technology for our classification stages is the ability of FL system to combine human expertise into computer-assisted decision making, facilitating more human-like decisions. FL is used in our proposed model to characterize XML message content sensitivity factors as fuzzy variables within each XML message, which determines the sensitivity level within the XML message to perform security measures on selected parts. The importance level rate for pre-selected nodes is a key factor to determine which encryption algorithm and keys are to be deployed. The importance level values are interpreted as "High", "Medium" or "Low".

During the fuzzification phase, element-wise encryption is performed on selected parts defined in the previous phase. Encryption type and key size is selected based on the "Importance Level" value. AES symmetric encryption with a 256-bit key size is deployed on tags with an "Importance Level" classified with a "High" value, and AES symmetric encryption with a 128-bit key size is deployed on tags with an "Importance

Level" classified with a "Medium" value. Tags with a "Low" value are forwarded to the message assembler without performing any kind of encryption.

4.2.1 Model Requirements

The system is designed to achieve a set of goals ensuring secure and efficient exchange of XML banking messages. The following requirements are needed to form the system core:

1) Messaging interface: Defines the syntax and semantics for the outgoing financial XML messages and ensures that our model understands them fully.
2) XML parser: Deciphers incoming XML messages to ensure they are fully understood prior to further processing.
3) Valid XML message: Submitted messages should be valid in terms of message structure whereby it represents the schema defined in the originating channel.
4) Stamped XML message: The first security layer which identifies that the incoming XML message is valid and from a trusted source. The stamp is added in the message header ensuring the originating channel, date of transmission, and service ID are all presented in a pre-defined sequence.
5) Communication Port: A dedicated communication port needs to be used in our XML submission, which is different from the one used in service messaging.
6) XML fuzzy classification characteristics: The 10 characteristics that define each transaction, each item should be available in the XML message; this will allow our system to build the classification criteria using our fuzzy classification phase.

7) Encryption algorithm: An encryption algorithm is needed to perform element-wise encryption. The AES encryption standard is being used in our model; we have chosen AES encryption over other algorithms for the following reasons:
    - Fast deployment for encryption/decryption processes in both software and hardware.
    - AES uses three key sizes: 128, 192, and 256 bits.
    - Advanced Encryption Standard not only assures security but also improves the performance in a variety of settings such as smartcards, and hardware implementations.
    - Currently there is no known non-brute-force direct attack against AES.
    - However, this can be replaced with other symmetric encryption standards.

8) Encryption Key Management: The encryption keys that are going to be used in our encryption phase must be managed.
9) D-H Key Exchange: The “public-key” or “asymmetric” cryptographic keys must be utilized.
10) Message assembler: To assemble different XML parts coming from different stages, encrypted or forwarded parts should be received and combined in one final message.

4.2.2 System Architecture and Design

Based on system requirements, the system architecture is illustrated in Figure 4.1. Major system components are described in details in figure 4.1. System modules act as

independent units whereby each unit can act as separate system; modules are combined to form SXMS

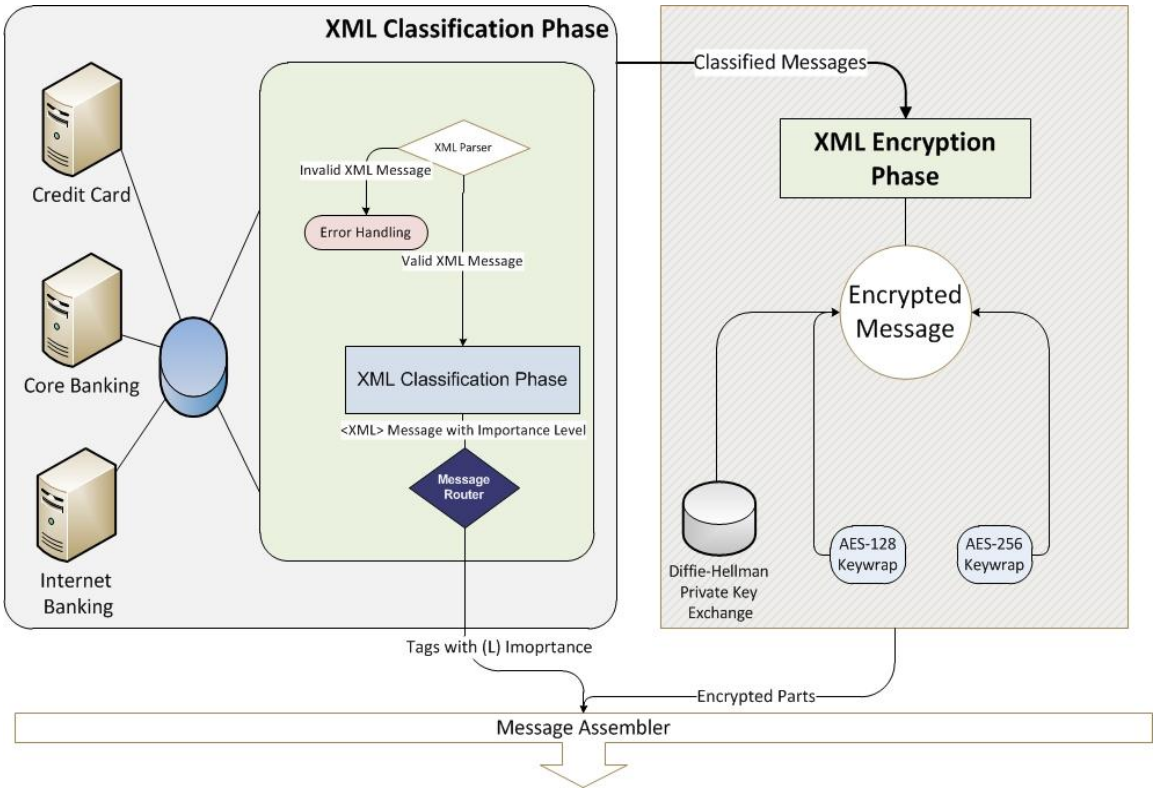

Figure 4.1: Main system components

As illustrated in Figure 4.1, the system architecture core is built based on two main modules forming the system core:

A. Fuzzy Classification Module: In our fuzzy classification module, we categorized 10 transaction characteristics into three different layers according to their type. The characteristics were chosen after exploring different experts' opinions and backgrounds, reviewing financial analysis tools, reviewing technical reports, researching different online and offline financial systems conducted within the financial institution, and performing a set of internal surveys among banking group heads. We categorized these 10 transaction characteristics extracted from the XML message into three layers

(Account Segment, Details Segment, and Environment Segment). Grouping will facilitate and simplify the process of fuzzy classification.

This phase performed a set of intelligent fuzzy classification techniques to assign a new value to an existing tag within each XML message. We called the tag “Importance Level”, and the main idea is distinguish which parts of the XML message is to be encrypted using element-wise AES asymmetric encryption, and which parts are to be forwarded directly to the message assembler without further processing. The key size for AES encryption is dependent on the “Importance Level” value assigned in this stage. The 10 characters are defined as follows:

1) **Transaction Amount**: Financial institutions set pre-defined transaction limits. The limits allow users to perform transactions with specified limits on a daily basis. The range of transaction limits is defined based on the local policy within each institution. Banks normally treat the transaction amount as an alert to any critical transaction; the amount is used in most banks to measure the weight of the total transaction performed. Source, destination, and amount all combine to act as an alert which is already pre-defined based on the bank's policy. Large transaction amounts will affect the importance of the transaction itself, which can be used in our model as a measurement item in our importance-level evaluation.
2) **Transaction Currency**: There is a well-defined list of allowed currencies that can be used online or offline. Each currency has its own set of risk variables depending on usage and importance. Foreign currency uses exchange rates, operational interference, and market value for the transaction the moment occurred. Banks treat each FX transaction with high importance, because it involves buying and selling with bank's rate. We

have used this factor in our importance evaluation.

3) **Account Type**: Accounts are segmented within each institution. Segmentation is performed to enable application of a set of internal rules on selected segments. Each segment has its own value and weight, for example corporate account segments are listed with high importance and priority because most of the transactions involve large volumes which can benefit the bank for each transaction. We used this factor due to its role in deciding the importance level for the whole transaction.
4) **Transaction Notes**: Exceptions are placed upon unusual activity on a specific account, and such exceptions will raise a flag in any transaction being processed to handle the exception before the process is completed. Having a flagged transaction will raise the importance level and trigger an alert to monitor that specific transaction due to its importance; we have used this factor to measure the importance level in terms of a transaction's critical weight.
5) **Profile ID**: A unique identifier for the destination account owner, the value is set during the system integration and profile creation process. Companies or individuals with custom profile IDs have a high potential to be monitored for transactions, and monitoring is based on the transaction amount after classifying each profile ID whereby a range of IDs are listed in the high importance zone, all after deploying a bank's methods and procedures.
6) **Account Tries**: How many times the account is used in the system; more usage means more trust whereby the history of the account is known and trusted. A historical log is kept and evaluated on a regular basis to confirm

trusted accounts and suspicious ones. The evaluation will result in a set of important ranges of trusted accounts to be used in the transaction evaluation and setting the importance level.

7) **Incorrect Password Tries**: The number of times users try to enter the password incorrectly to complete the financial transaction. This factor adds a slight level of importance to each transaction, with a high rate of incorrect tries giving an indication of high importance.
8) **Time Spent on the Service**: The time spent navigating the service before performing the transaction. The time range is set based on the bank's policy, taking into consideration peak hours. This factor considers technical factors to measure the importance level of the transaction which is based on non-financial elements.
9) **Daily Transactions**: How many transactions are performed before the financial transaction is carried out. The number of daily transactions puts a weight on the overall importance level for the transaction itself, whereby the number of transactions to be performed is set based on the bank's policy within the allowed ranges.
10) **Transaction Time**: The financial day is categorized in three periods: peak period, normal hours, and dead zone. Periods are defined separately by the financial institution based on local policy and the historical transactions range. Each period has its own value which adds a level of importance and how the occurrence of any transaction is affected by the time of occurrence. Ranges are set to weigh an importance level when the transaction is performed.

B. Encryption Module: This module operates by performing an element-wise encryption using the AES encryption algorithm. Element-wise encryption is performed on selected portions of an XML document and their corresponding nodes defined previously in the fuzzy classification module. The encryption process can be applied to any number of elements whereby it is encoded using base64. Two key sizes are being used in this module, a 256-bit key and a 128-bit key, and usage depends on the "Importance Level" tag value classified earlier. Attributes with a value of "High" are encrypted using a 256-bit key size, attributes with a value of "Medium" are encrypted using a 128-bit key size, and finally the attributes with a value of "Low" are forwarded directly to the message composition stage without performing any kind of encryption.

C. Message Assembly Module: This module is responsible for gathering all pieces together, all in sequence of arrival. Encrypted and non-encrypted parts are being assembled for final submission to the message destination channel.

In chapter 5 we will test and evaluate above components. While we design the system, we took in consideration the following points:

- Ability to test against main requirements, each requirement should be tested with ease.
- System should be structured and well-defined. System code should be readable and easy to understand.
- Ability to reuse components, system design should be reusable.

4.3 Fuzzy Classification Methodology

Methodology main technique involves the fuzzification of input variables based on 10 characteristics extracted from the XML message, Figure 4.2 illustrates the whole fuzzy

classification process and internal stages, rule evaluation, aggregation of the rule outputs, and defuzzification phases are displayed.

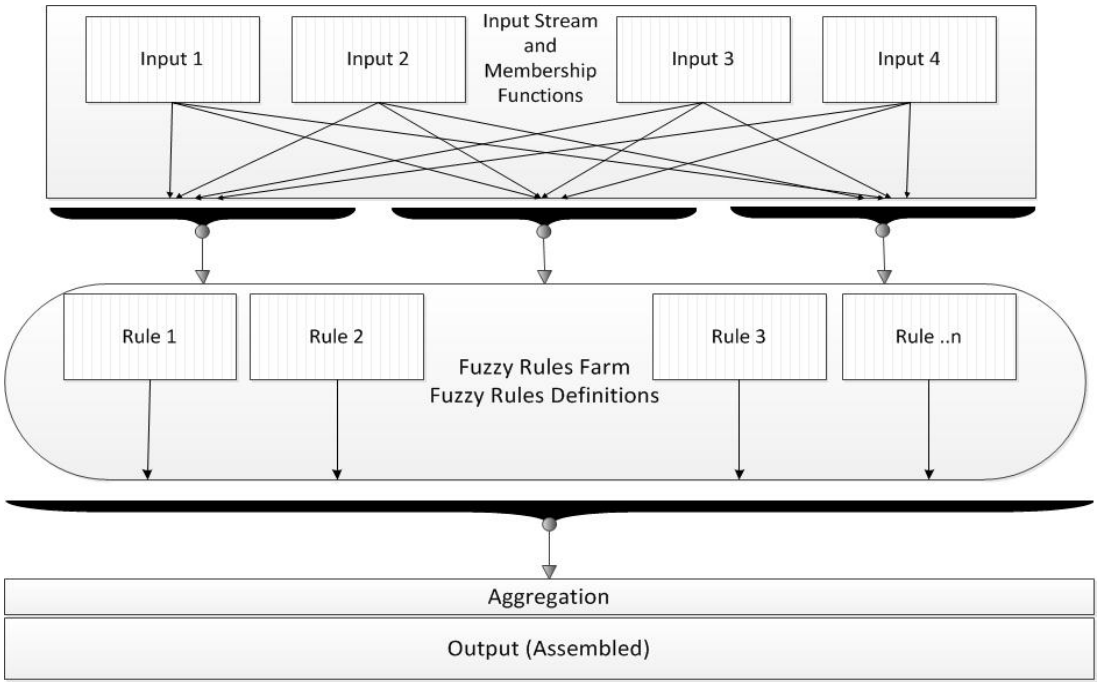

Figure 4.2: Fuzzy Inference System

The fuzzy inference system involves four phases. A comprehensive description for each phase will be explained in order to understand main functionality behind each phase. Connection between each phase is described as well.

4.3.1 Fuzzify Input Stage

The first step is to take the inputs extracted from the 10 characteristics within each XML message and determine the degree to which they belong to each of the appropriate fuzzy sets via membership functions. The input is always a crisp numerical value limited to the universe of discourse of the input variable (in our case it is an interval between 0 and 10) and the output is a fuzzy degree of membership in the qualifying linguistic set (interval between 0 and 1). Fuzzification of the input amounts to a function evaluation. A decision mechanism for identifying importance-level values within each XML message will be provided.

We built this stage on three rules, and each of the rules depends on resolving the inputs into a number of different fuzzy linguistic sets: the factor is non-sensitive, the factor is normal, and the factor is sensitive. Before the rules can be evaluated, the inputs must be fuzzified according to each of these linguistic sets. For example, the transaction amount can range from "Non-sensitive" to "Sensitive" with other values being taken into account. The degree of membership decides the degree of belongingness of the values of variables to any class. We have designed a membership function for each transaction's characteristic indicator, which clearly defines how each input is mapped to a membership value between [0, 1]. Linguistic values are assigned to each transaction factor as non-sensitive, normal, and sensitive and for the final layer rate as High, Medium, and Low . Figure 4.3 represents an example of the linguistic descriptors used to illustrate one of the key transaction characteristics, transaction amount, with a plot of the fuzzy membership functions. The range of possible values for the corresponding key importance characteristic (Non-Sensitive, Normal, and Sensitive) is represented in x-axis. The linguistic descriptor represents the degree to which a value for the importance level characteristic and it is represented in y-axis.

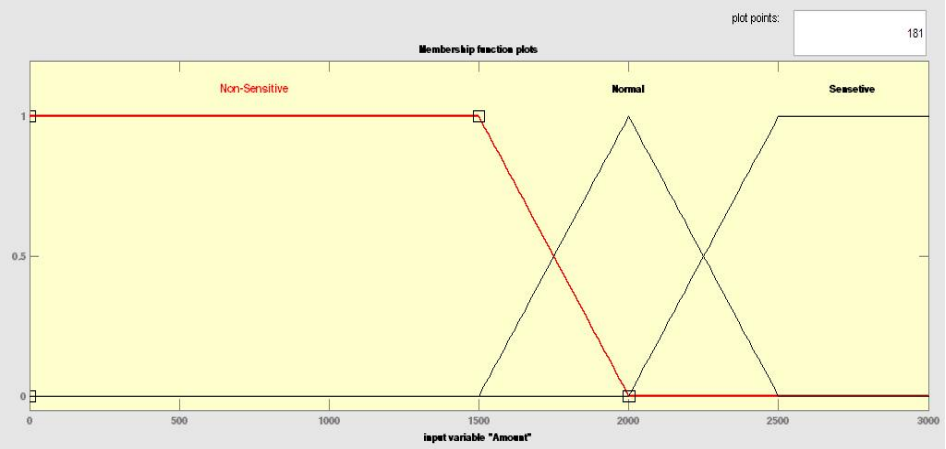

Figure 4.3: Input variable for transaction amount factor

Transaction Amount (Non-Sensitive, Normal, Sensitive)

Linguistic Variable: Transaction Amount value

Linguistic Value	Range (in Numbers)

Sensitive	[2000, 2500, 3000, 3000]
Normal	[1500, 2000, 2500]
Non-Sensitive	[0, 0, 1500, 2000]

4.3.2 Rule Evaluation Stage

The second step is to take the fuzzified inputs and apply them to the antecedents of the fuzzy rules. If a given fuzzy rule has multiple antecedents, the fuzzy operator (AND or OR) is used to obtain a single number that represents the result of the antecedent evaluation. This number (truth value) is applied to the consequence membership function. Then we specify how the importance level probability is different as a function of the main importance level characteristic factors. A set of fuzzy rules are represented in form of (IF-THEN) statements and provided by experts; these statements connect to the importance level probability of different levels of importance characteristic factors based on previous experience.

4.3.3 Aggregation of the Rule Output Stage

This is the process of unification of the outputs of all the rules. It means we take the membership functions of all rules' consequents previously scaled and then combine them in one single fuzzy set. The list of scaled resulting membership functions is considered the aggregation process input, one fuzzy set for each output variable considered as an output.

4.3.4 Defuzzification Stage

This is the last step in the fuzzy inference model; the final output has to be a crisp number. The input for the defuzzification process is the aggregate output fuzzy set and the output is a number. We have conducted the Centroid technique in the defuzzification stage (Sugeno, 1985). This technique helps us to find the point where a vertical line would hit the aggregate set into two equal masses. The final output is the

importance level rate which is defined in fuzzy sets (“High”, “Medium”, and “Low”). Figure 4.4 illustrates the final output ranges.

Linguistic Variable: Importance Level

Linguistic Value	Range (In Numbers)
High	[6, 8, 10, 10]
Medium	[3, 5, 7]
Low	[0, 0, 2, 4]

High: if the importance level of the tag is considered “High”, this will impact upon the overall transaction within the XML message.

Medium: this might affect the overall transaction rate in some way; the content could be important for further attention.

Low: with a low importance level, no impact is to be considered in the XML transaction.

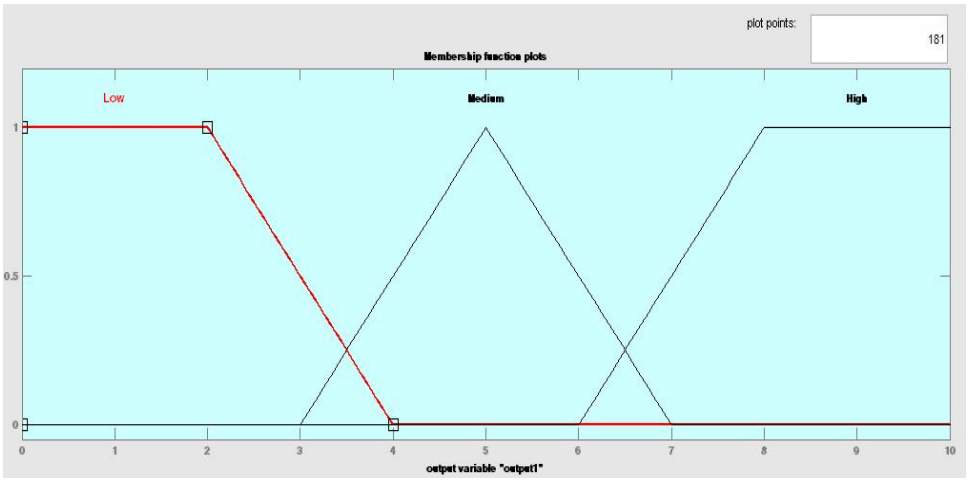

Figure 4.4: Output variable and ranges

4.4 Fuzzy Classification Model

In our fuzzy classification model, we categorize the 10 transaction field characteristics into three different layers (Transaction Layer, Details Layer, and Environment Layer) based on their nature among overall transaction type whereby each element within the layer has its own importance measures. To improve the final importance level rate

(fuzzy output), we have conducted a layering process for these features. Table 4.1 illustrates the grouping mechanism and layering details based on XML message content.

Table 4.1: XML message content components and layers

Components	Sequence	Layer Name	Layer
Transaction Amount	1	**Account Segment**	Layer 1
Transaction Currency	2		
Account Type	3		
Components	**Sequence**	**Layer Name**	**Layer**
Account Tries	4	**Details Segment**	Layer 2
Transaction Notes	5		
Profile ID	6		
Incorrect Password Tries	7		
Components	**Sequence**	**Layer Name**	**Layer**
Daily Transactions	8	**Environment Segment**	Layer 3
Transaction Time	9		
Time Spent on Service	10		

Figure 4.5 illustrates architecture design of the fuzzy inference classification model. As seen in the figure, each layer has an output which indicated that the tag rate that depends on the evaluation (fuzzy outputs) of the layer components.

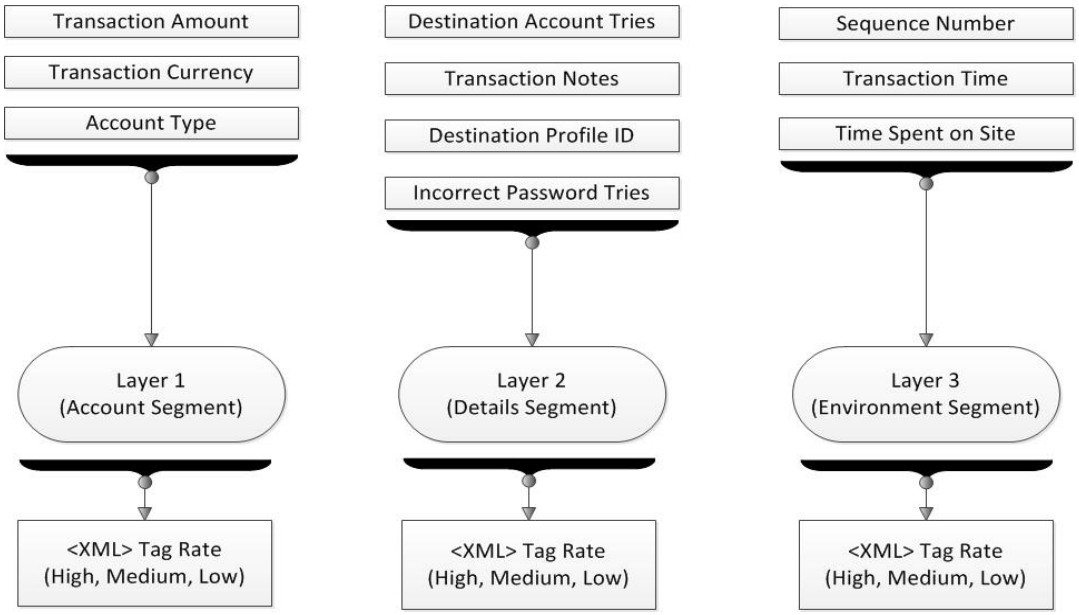

Figure 4.5: Classification Architecture of the importance level fuzzy mode (TAG Classification)

Our fuzzy classification model is performed based on three layers: Transaction Layer, Details Layer, and Environment Layer. Each layer has a set of components that are distributed based on the component nature and usage; the total number of components is 10 and they are distributed among the three layers as follows: transaction amount, transaction currency, and account type are fitted in layer 1 (account segment layer); account tries, transaction notes, profile ID, and incorrect password tries are fitted in layer 2 (details segment layer); and finally, the daily transactions, transaction time, and time spent on site are fitted in layer 3 (environment segment layer).

4.5 Fuzzy Rule Base and Layers Categorization

4.5.1 Rule Base for Layer 1

The rule base for layer 1 has three inputs and one output. The rule has all the "IF-THEN" conditions of the system. For each entry of the rule base, each component is assumed to be one of the three values and each criterion has three components. This means that rule base 1 contains 27 entries (33). The output of rule base 1 is one of the importance level fuzzy sets (High, Medium, or Low) representing the account segment layer importance level. Table 4.2 illustrates a sample structure along with sample entries of rule base 1 for layer 1.

Table 4.2: Rule base 1 for the account segment layer – layer 1

Transaction Amount	Transaction Currency	Account Type	Account Layer Importance Level Rate
Non-Sensitive	Non-Sensitive	Non-Sensitive	**Low**
Non-Sensitive	Non-Sensitive	Normal	**Low**
Non-Sensitive	Sensitive	Non-Sensitive	**Low**
Normal	Normal	Normal	**Medium**
Normal	Non-Sensitive	Sensitive	**Medium**
Normal	Sensitive	Normal	**Medium**
Sensitive	Non-Sensitive	Sensitive	**High**
Sensitive	Non-Sensitive	Non-Sensitive	**Low**
Sensitive	Sensitive	Non-Sensitive	**High**

The system structure for the account segment layer is the result of joining the three basic components (transaction amount, transaction currency, and account type), which generates the layer importance level. Figure 4.6 and Figure 4.7 illustrate the structure of the system and the three-dimensional surface structure, respectively. MATLAB R2010R has been used in the process.

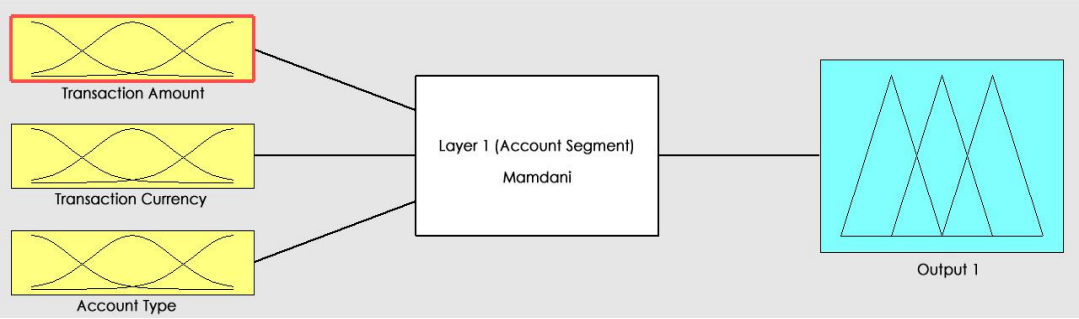

Figure 4.6: Layer 1 system structure (inputs and outputs)

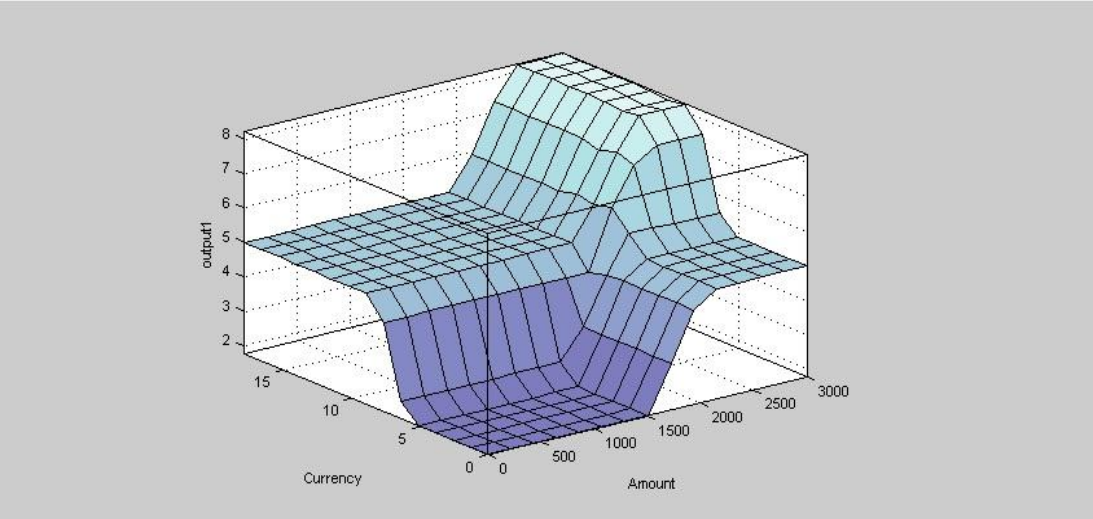

Figure 4.7: Surface structure in a three-dimensional view for layer 1

4.5.2 Rule Base for Layer 2

The rule base for layer 2 has four inputs and one output. The rule has all the "IF-THEN" conditions of the system. For each entry of the rule base, each component is assumed to be one of the three values and each criterion has four components. This means the rule base for layer 2 contains 81 entries (34).

The output of rule base is one of the importance level fuzzy sets (High, Medium, or Low) representing the details segment layer importance level. Table 4.3 illustrates a sample structure along with sample entries of rule base for layer 2.

Table 4.3: Rule base 1 for the details segment layer – layer 2

Transaction Notes	Profile ID	Account Tries	Incorrect Password Tries	Details Layer Importance Level Rate
Non-Sensitive	Sensitive	Sensitive	Sensitive	**High**
Non-Sensitive	Normal	Sensitive	Sensitive	**High**
Non-Sensitive	Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Normal	Sensitive	Non-Sensitive	Normal	**Medium**
Normal	Sensitive	Normal	Non-Sensitive	**Medium**
Normal	Non-Sensitive	Normal	Non-Sensitive	**Low**
Sensitive	Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Sensitive	Non-Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Sensitive	Non-Sensitive	Non-Sensitive	Sensitive	**High**

The details segment layer system structure is the result of joining the four basic components (transaction notes, profile ID, account tries, and incorrect password tries), which generates the layer importance level. Figure 4.8 and Figure 4.9 illustrate the structure of the system and three-dimensional surface structure, respectively. MATLAB R2010R has been used in the process.

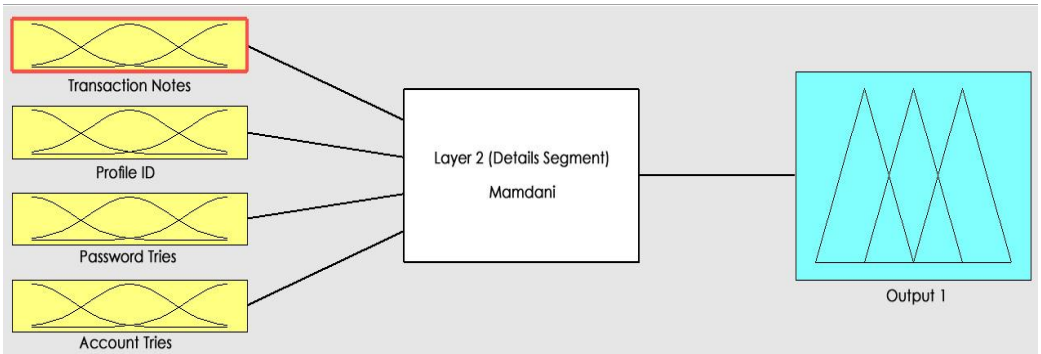

Figure 4.8: Layer 2 system structure (inputs and output)

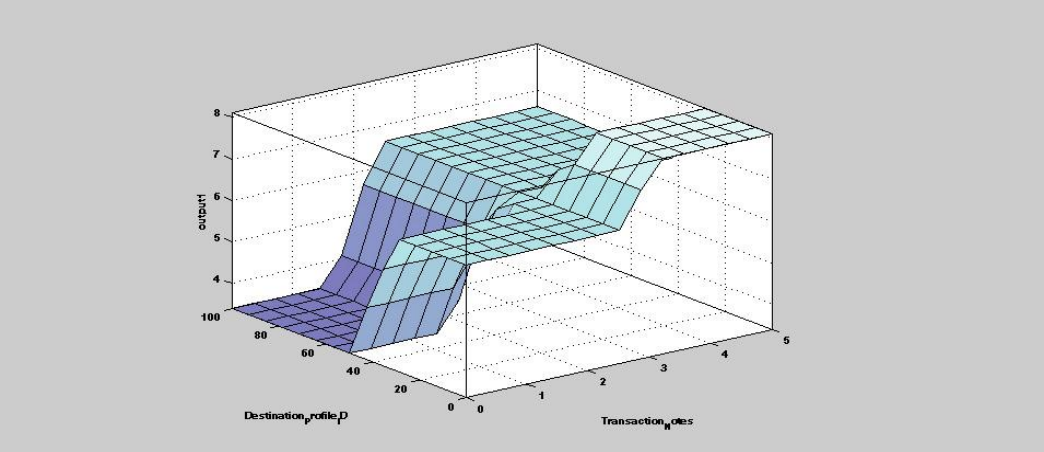

Figure 4.9: Surface structure in a three-dimensional view for layer 2

4.5.3 Rule Base for Layer 3

The rule base for layer 3 has three inputs and one output. All of the "IF-THEN" conditions of the system reside within the rule. Each component of each entry of the rule base is assumed to have one of the three values. This means the rule base has 27 entries (3^3). The rule base output is one of the importance level fuzzy sets (High, Medium, or Low) representing the environment segment layer importance level. Table 4.4 illustrates a sample structure and sample entries of the rule base for layer 3.

Table 4.4: Rule base 1 for the environment segment layer – layer 3

Time on Site	Daily Transactions	Transaction Time	Environment Layer Importance Level Rate
Non-Sensitive	Normal	Sensitive	**High**
Non-Sensitive	Sensitive	Sensitive	**High**
Non-Sensitive	Sensitive	Normal	**Medium**
Normal	Non-Sensitive	Normal	**Medium**
Normal	Sensitive	Non-Sensitive	**Medium**
Normal	Non-Sensitive	Non-Sensitive	**Low**
Sensitive	Normal	Sensitive	**High**
Sensitive	Sensitive	Sensitive	**High**
Sensitive	Non-Sensitive	Sensitive	**High**

The system structure for the environment segment layer is the result of joining the three basic components (Time on Service, Daily Transactions, and Transaction Time), which

generates the layer importance level. Figure 4.10 and Figure 4.11 illustrate the structure of the system and the three-dimensional surface structure. MATLAB R2010R has been used in the process.

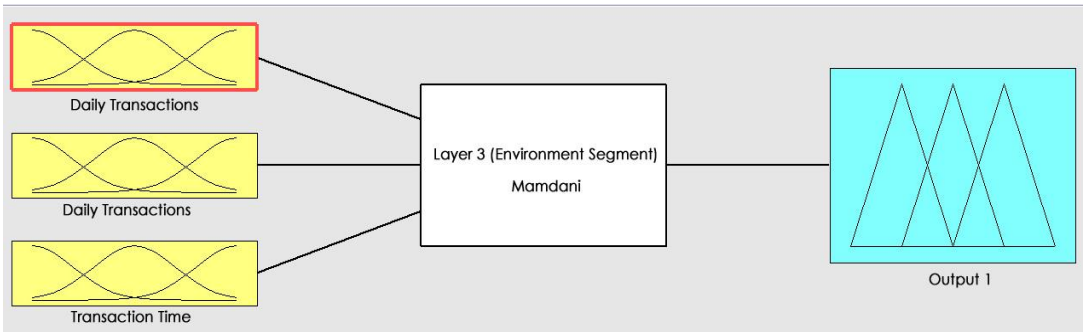

Figure 4.10: Layer 3 system structure (inputs and output)

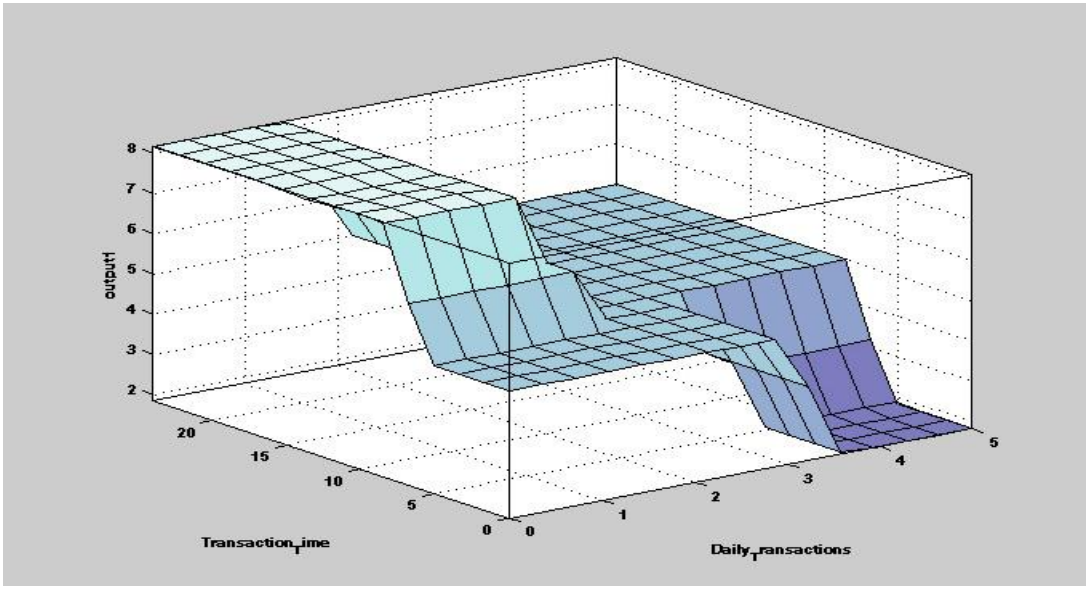

Figure 4.11: Surface structure in a three-dimensional view for layer 3

4.6 Encryption Model

4.6.1 Element-wise Encryption

In the previous step, we conducted fuzzy classification techniques on the XML messages in order to classify the included tags. The main idea of the process is to distinguish between the sensitive parts that require the deployment of security measures and the other parts that need no processing. Therefore the second phase in our model is

to apply element-wise encryption on the classified parts within each XML message. Element-wise encryption was first introduced by (Maruyama & Imamura, 2000). Figure 4.12 illustrates the system structure when deploying element-wise encryption on selected parts.

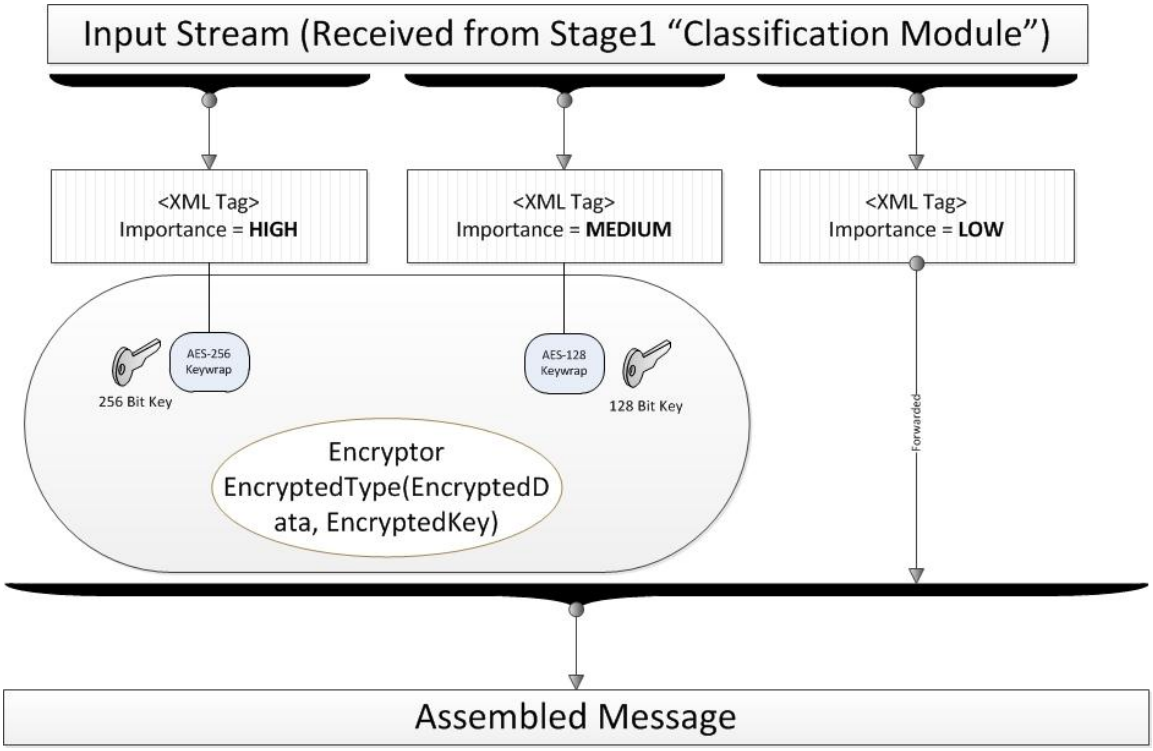

Figure 4.12: Encryption module architecture and design

As displayed in Figure 4.12, the tags with an importance level of "High" or "Medium" are being encrypted using different keys depending on the importance level value, and tags with an importance level of "Low" are being forwarded without any need of encryption.

4.6.1.1 Element-wise Encryption Standard

In order to deploy element-wise encryption on selected parts within XML message, the following list of requirements need to be fulfilled first:

1. Element-wise confidentiality: Elements within the XML document should be able to be encrypted. Elements might enclose different types; the remaining parts of the XML document should remain in plaintext.
2. The encrypted document should be well-formed: the resulting document (after encryption has been deployed on the selected parts) has to be a well-formed XML document. Encryption can be nested within the document.
3. The same set of basic information items must be exactly provided as the original document (when decrypting an encrypted document). Comments can be ignored. CDATA and entity references sections are not preserved.
4. Independence from encryption algorithm: symmetric and asymmetric encryption must be supported, whereby the encrypted elements syntax should be isolated from the encryption algorithm used.
5. Flexible key delivery mechanism: Two main key exchanges should be supported, the first is the certificate-based key exchange (encryption key is embedded in the syntax) and the second is the out-of-band key exchange (no key elements are embedded in the syntax).
6. Content independency: the outer text should not play a role in decrypting the encrypted parts. For instance, the character encoding for the outer context might be changed without affecting the encrypted document content.
7. Verification and validation by receiver: the incoming decrypted XML document should be validated by the receiver.
8. Intermediary validation: this refers to the ability of an intermediary to validate the encrypted document without the decryption keys and without the need to decrypt the element. No need to validate the included content models, but the validation of any outer text.

9. Content model confidentiality: it should be feasible to maintain the content model of the encrypted data confidential. By knowing the content model of the element, there is a good chance to have an idea of which element to attack (if the XML document has multiple encrypted elements).

Figure 4.13 illustrates a sample XML message after the fuzzy classification phase is performed, showing the “Importance Level” attribute values.

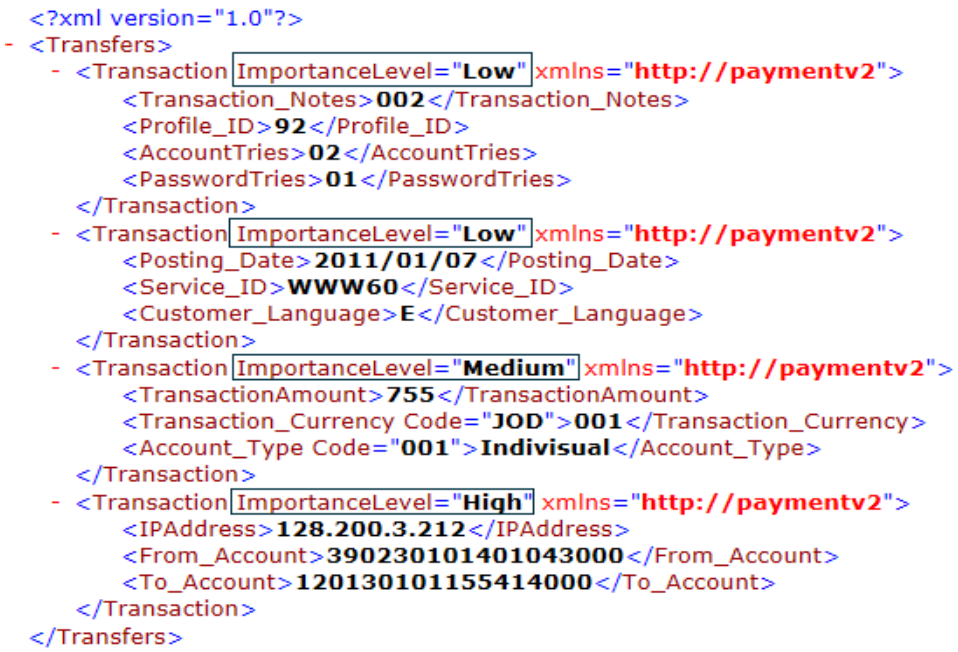

Figure 4.13: Classified XML message to be encrypted using element-wise encryption

Figure 4.14 illustrates the same XML message after element-wise encryption is deployed. The tag attribute marked with a “Low” importance level will have no encryption deployed, while the tags with an “Importance Level” value of “High” or “Medium” will be encrypted using the AES encryption algorithm.

Figure 4.14: XML message after deploying element-wise encryption on selected parts

4.6.1.2 Design Consideration

All of the fuzzy rules employed in our fuzzy classification model were obtained based on our banking and financial expertise, combined with financial experts knowledge and supported by a set of experimental stages empowered by real life examples. Next we will show all fuzzy rules for all XML content characteristics.

1. Element serialization: XML documents (with sub-elements) need to be converted into a byte sequence before encryption because the AES encryption algorithm (which is used in our model to encrypt the selected tags) treats the inputs as a byte sequence. Our conversion tries to minimize the loss of any non-essential information during the encryption process, like DOCTYPE declaration, use of namespace prefixes, and character encoding of the document. In our encryption phase we preserve the XML information set during the serialization operation by using canonical XML (W3C, BOYER, EASTLAKE, & REAGLE,

2002). We use C14N because of the implementation language independency. Encryption and decryption processes can be achieved in normal ways once the element is converted into byte array.

2. Packaging encrypted elements in XML: when inserting the encrypted elements (binary data) into an XML document, Base64 is used as a text encoding method. Additionally, it is necessary to attach additional information for the decryption process like initialization vector, encryption algorithm, and keys. Usage of an existing cryptographic message would alleviate the burden of having security flaws.

   The cryptographic message format that is used in S/MIME (PKCS7/CMS) considered one of the most popular formats. In S/MIME, the encrypted process is performed by packing the cipher text alongside related information in a single MIME entity (media type should be application/pkcs7-mime). It can be included in an XML document in one of the standard ways to embed MIME objects.

Child nodes related to the parent tag are also encrypted using the same level of encryption. Child tags' behaviour is taken from the parent "Importance Level" value. In Figure 4.14, tags (Transaction Amount, Currency, and Account Type) are encrypted using AES encryption with a 128-bit key size as per their parent "Account" layer. Basically we inherit the encryption behaviour from parent to child as per our categorization process.

4.6.1.3 XML Message Schemas

Syntax: Schema for XML core encryption, Figures 4.15, 4.16, 4.17, 4.18, 4.19, and 4.20 illustrate XML schema for core encryption and other sub elements.

Syntax: Schema for XML core encryption

```
<?xml version="1.0"?>
<schema targetNamespace="http://www.w3.org/xmlenc">
        xmlns="http://www.w3.org/1999/XMLSchema"
        xmlns:xenc="http://www.w3.org/xmlenc">

<simpleType name="base64-encoded-binary" base="binary">
  <encoding value="base64"/>
</simpleType>
```

Figure 4.15: XML Schema for core encryption

Syntax: Schema for the *EncryptionInfos* Element

```
<element name="EncryptionInfos">
  <complexType>
    <choice minOccurs="1" maxOccurs="unbounded">
      <element ref="xenc:EnvelopeInfo" />
      <element ref="xenc:EncryptionInfo" />
    </choice>
  </complexType>
</element>
```

Figure 4.16: XML schema for *EncryptionInfos* element

Syntax: Schema for the *KeyValue* Element

```
<element name="KeyValue">
  <complexType>
    <choice>
      <any minOccurs="1" maxOccurs="unbounded" />
      <element ref="xenc:AESKeyValue" />
    </choice>
  </complexType>
</element>
```

Figure 4.17: XML schema for *KeyValue* element

Syntax: Schema for the *ContentEncryptionMethod* Element

```
<element name="ContentEncryptionMethod">
  <complexType>
    <any minOccurs="0" maxOccurs="unbounded" />
    <attribute name="Algorithm" type="uriReference"
use="required" />
  </complexType>
</element>
```

Figure 4.18: XML schema for the *ContentEncryptionMethod* element

Syntax: Schema for the *Reference* Element

```
<element name="Reference">
  <complexType>
    <element ref="xenc:CanonicalizationMethod" minOccurs="0"
maxOccurs="1" />
    <attribute name="URI" type="uriReference" use="optional" />
    <attribute name="Id" type="ID" use="optional" />
    <attribute name="MimeType" type="string" use="optional" />
  </complexType>
</element>
```

Figure 4.19: XML schema for the *Reference* element

Syntax: Schema for the *EncryptedContent* Element

```
<element name="EncryptedContent">
<complexType base="xenc: base64-encoded-binary"
derivedBy="extension">
<attribute name="Id" type="ID" use="optional" />
<attribute name="URI" type="uriReference" use="optional" />
</complexType>
</element>
```

Figure 4.20: XML schema for the *EncryptedContent* element

4.6.2 Diffie–Hellman key exchange

The DH Algorithm, first introduced by (Diffie & Hellman, 1976), was the first system to utilize "Public-key" cryptographic keys. We used the DH system because of its ability to manage public keys easily. As we use symmetric encryption "AES" with different key sizes, both sides of the communication must have identical keys. Securing the submitted keys has been an issue, so we use our model (symmetric-based encryption system) to encrypt the XML documents and DH system (asymmetric-based system) to encrypt the symmetric keys for distribution. In brief, we use the DH algorithm for public key exchange.

4.6.2.1 D-H Process:

We use Diffie-Hellman to secure exchanged keys, not the actual content which is already encrypted using our element-wise encryption module. To achieve the efficient exchange, we create a shared secret key, which we call "Key Encryption Key - KEK",

between the two entities (the sending channel and the receiving channel). Later the shared secret key encrypts the symmetric key for secure submission. The resulting symmetric key is called the traffic encryption key (TEK) for the data encryption key (DEK). This means KEK secures the delivery of TEK, while TEK provides the secure delivery of the XML document itself.

The process starts just after the generation of private keys by each side (sending and receiving channels). Each side then generates a public key, which is derived from the private key because both keys are mathematically linked. Then the two entities (sender channel and receiver channel) exchange their public keys. Each side now has its own private key and the other entity public key.

During the key exchange process, the DH protocol generates "shared secret" keys that are identical and shared by both entities. A set of mathematical operations are performed against the sender's private key and the receiver's public key which generate a value. The reverse process (performing mathematical operations against the sender's public key and the receiver's private key) is performed to generate another value. The two values should be identical. The combination of the two values is called the "Shared Secret" which encrypts the information between the two entities. In our model, the shared secret of the Diffie-Hellman protocol encrypts a symmetric key for our AES encryption algorithm, transmits it securely, and the distant end decrypts it with the shared secret. Figure 4.21 shows how the DH process operates. The sender is the one who actually generates and transmits the symmetric key in most cases. However, it can be handled by both sender and receiver.

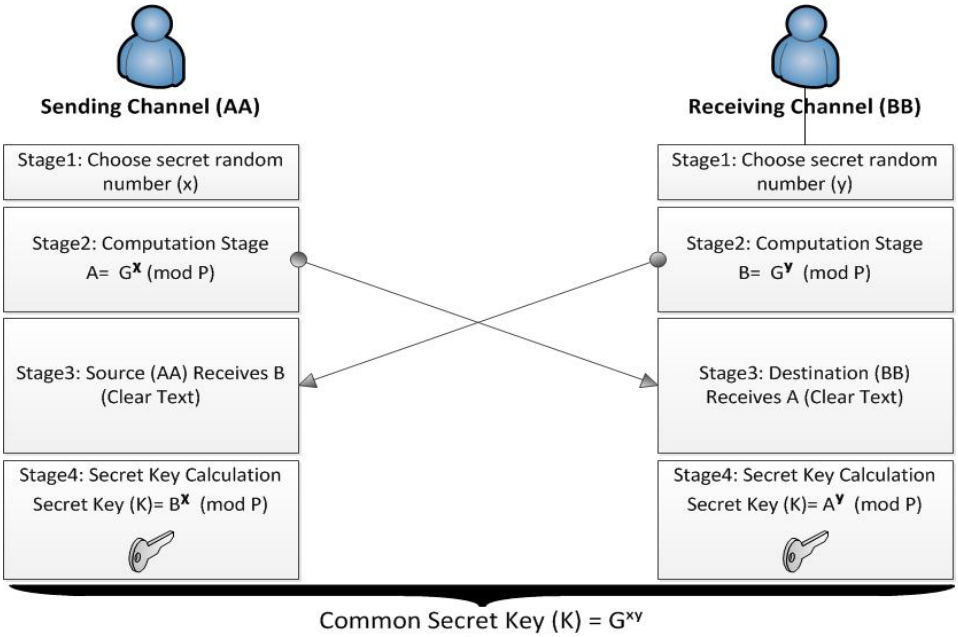

Figure 4.21: Key exchange using the DH method

As seen in figure 4.21, we require two large numbers, one prime (P), and (G), which is a primitive root of (P). P should be at least 512 bits. In stage 2 we compute public values for both sender and receiver (A, B) using:

$A = G^{X}(mod\ p)$ Where X is the generated number by sender in stage 1

$B = G^{Y}(mod\ p)$ Where Y is the generated number by receiver in stage 1

Now we compute the shared and secret keys by applying:

$K_a = B^x\ (mod\ p)$ Where K_a is the secret key for sender

$K_b = A^y\ (mod\ p)$ Where K_b is the secret key for receiver

```
<element name="DHKeyValue" type="xenc:DHKeyValueType"/>
<complexType name="DHKeyValueType">
   <sequence>
      <sequence minOccurs="0">
         <element name="P" type="ds:CryptoBinary"/>
         <element name="Q" type="ds:CryptoBinary"/>
         <element name="Generator"type="ds:CryptoBinary"/>
      </sequence>
      <element name="Public" type="ds:CryptoBinary"/>
      <sequence minOccurs="0">
         <element name="seed" type="ds:CryptoBinary"/>
         <element name="pgenCounter" type="ds:CryptoBinary"/>
      </sequence&gy;
   </sequence>
</complexType>
```

Figure 4.22 represents the DH key agreement schema.

4.7 Message Utilization

Message security is the main concern to be handled in our model. However, deploying our framework will enable us to utilize outgoing messages and save resources by reducing outgoing messages. Figure 4.23 illustrates how the outgoing messages are utilized whereby only essential parts are being secured; other parts are just forwarded to the message assembler without the need for any type of processing.

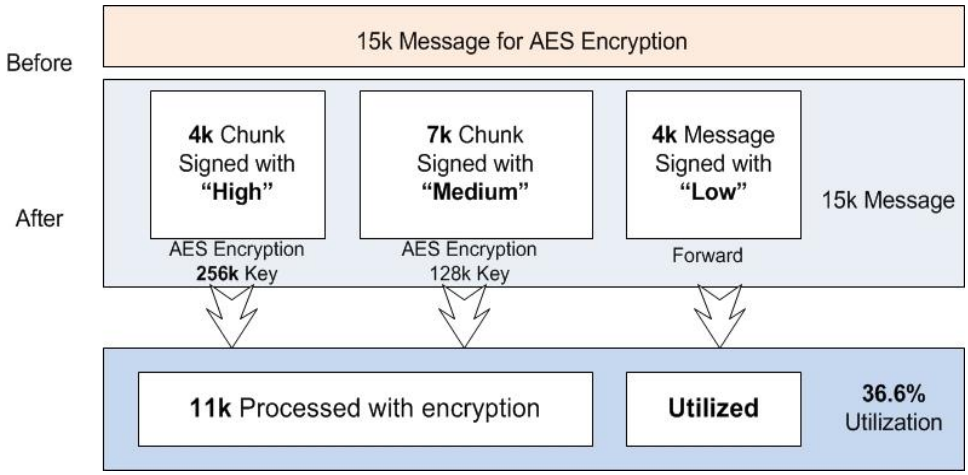

Figure 4.23: Message utilization

Once the utilization process is completed, messages are ready for final submission to the selected destination.

4.8 Chapter Summary

A novel approach for securing financial XML messages using intelligent mining fuzzy classification techniques has been proposed. This approach will secure financial XML messages using element-wise XML encryption, fuzzy classification techniques that allow us to classify XML messages on the fly without the need to define a set of rules prior to the process.

Mining fuzzy classification techniques have been used to evaluate and measure the data sensitivity level within each XML message to find a degree of sensitivity for each tag in the XML message. The mining fuzzy classification process allowed us to assign a value to a new attribute added to the parent XML nodes. A value is determined by applying a set of fuzzy classification processes based on Mamdani inference (Mamdani & Assilian, 1975). A new value has been used to determine which type of encryption algorithm is being performed on selected tags, allowing us to secure only the needed parts within each message rather than encrypting the whole message. XML encryption is based on W3C XML recommendations. Nodes that are assigned an importance level value of "High" will be encrypted using the AES encryption algorithm with a 256-bit key size to ensure maximum security. Nodes that are assigned an importance level value of "Medium" will be encrypted using the AES encryption algorithm with a 128-bit key size. An implementation was performed on a real-life environment using online banking systems to demonstrate its flexibility, feasibility, and functionality. There are many possible directions for future work which will be discussed in Chapter 5.

Chapter 5

SXMS Model Implementation and Testing

5.1 Introduction

Chapter 5 presents the main system components, system functionalities, and implementation tools that have been used to evaluate our secure XML management model. We also illustrate the detailed testing of SXMS model in general and the testing of the main parts of the SXMS model in specific. The testing strategy will involve testing each stage on its own. Additionally, we present the XML sample set extracted from our collected financial messages which have been used in our testing and evaluation stages. Finally, we summarize and conclude the chapter.

5.2 Development Architecture and Used Tools

5.2.1 Used Tools

System designed to achieve set of goals ensuring secure and efficient exchange of XML financial messages among different systems. Following measures are key factors in system design:

1- Technical Computing (MATLAB R2010a): Since we are performing fuzzy classification, we require a high-performance language for technical computing. It has the capability to integrate computation, visualization, and plotting of

functions and data. Also it allows interfacing with other applications written in other programming languages such as Java, C, and C++. In our case we require the integration with our Java application that is used to encrypt classified documents. Solutions are presented in an easy to understand and familiar mathematical notation.

2- Stylus Studio 2011 XML Enterprise Suite: Dealing with XML documents is the core of our model, since the files are in XML format we require an easy way to edit and manipulate XML files, XML schemas, and DTD codes. This tool allows us to check the validity of XML files, XML well-formedness checking, and parsing as well. Also we require the ability to edit XML schemas, DTD with inline capability.

3- Netbeans IDE 7.0: Our fuzzy classification module is built using Java J2EE, Netbeans Integrated development environment (IDE) is a platform framework for Java applications. The NetBeans Platform is a reusable framework for simplifying the development of Java Swing desktop applications. The NetBeans IDE bundle for Java SE contains what is needed to start developing NetBeans plugins and NetBeans Platform based applications.

4- Microsoft Office Excel is used as a graphical and chart presentation, as we will compare models and present results in graphical presentation. Excel allows us to have a clear and easy to understand presentation with measures presented on the graph.

5.2.2 Development Architecture

Our proposed secure XML management system has the ability to extract all of the 10 transaction characteristics and patterns for each XML transaction. Our fuzzy classification module main functionality is to classify XML messages based on these

characteristics and by deploying Mamdani fuzzy system. Once classified, the fuzzy classification module will assign a value to the "Importance Level" tag within the messages. Classified messages are being processed by our encryption module to perform element-wise encryption on selected classified parts.

System design includes three primary parts all of which signifies a crucial element of the requirements need. Those parts are:

- Parsing incoming XML messages: to check for validity and well-formedness by using XML Sax parser. In case of any issues, messages should be handled by error handler component for either return an error code or halt the operation.
- Classifying valid XML messages: fuzzy classification is done by using the fuzzification of input variables that is based on 10 characteristics extracted from the XML message.
- Routing messages: messages are being routed either to final message assembler or to the XML encryption phase.
- Encrypting incoming messages where "Importance Level" tag is labelled with "High" or "Medium": Tags with "Importance Level" set to "High" are being encrypted using 256-bit key, tags with "Importance Level" set to "Medium" are being encrypted using 128-bit key.
- Private Key Exchange: using Diffie-Hellman private key exchange to exchange keys between sender and receiver.

In this chapter, a set of testing and evaluation for the above listed components will be performed. During our design and development we took the following measures into consideration:

Reusability: ability to reuse included components or the system in complete.

Testability: ability to test the system against requirements.

Structured: ability to read code and structure easily.

Figure 5.1 presents the development architecture and design.

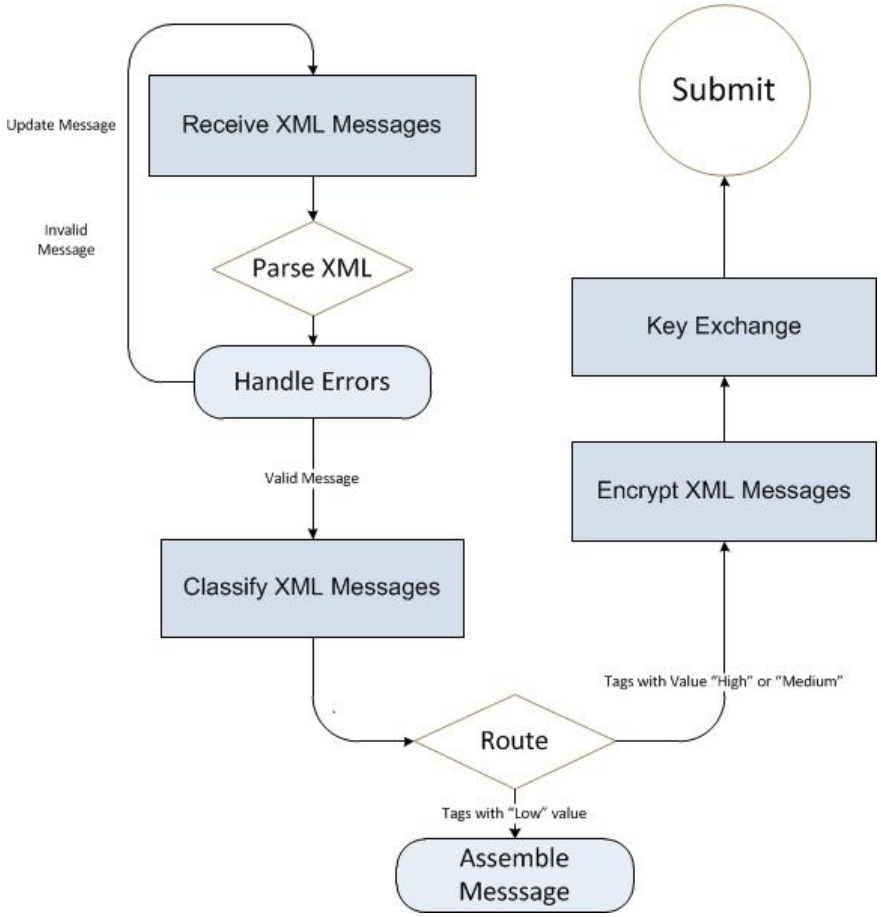

Figure 5.1: Model development architecture

5.3 System Implementation

5.3.1 SXMS Implementation Requirements

Table 5.1 represents the minimum software and hardware requirements needed for NetBeans IDE 7.0 and Stylus Studio 2011 XML Enterprise Suite to perform system testing and implementation.

Table 5.1: Minimum hardware requirements to run main tools

Operating System	Microsoft Windows 7 – Ultimate edition, 64-bit / SP1
Environment	Java Development Kit JDK v7.0
Processor	PC / Intel based technology 2.0 MHz or faster processors
Memory	512 MB of RAM
Disk Space	750 MB of free disk space
Display	1280x 768 px
Drive	DVD Drive

5.3.2 SXMS Implementation Process

Based on approaches in Chapter 4, SXMS fuzzy classification module generates on-the-fly fuzzy classification value; this value is assigned to the Importance Level attribute in each XML message. The major steps for the fuzzy classification module are shown in Figure 5.2.

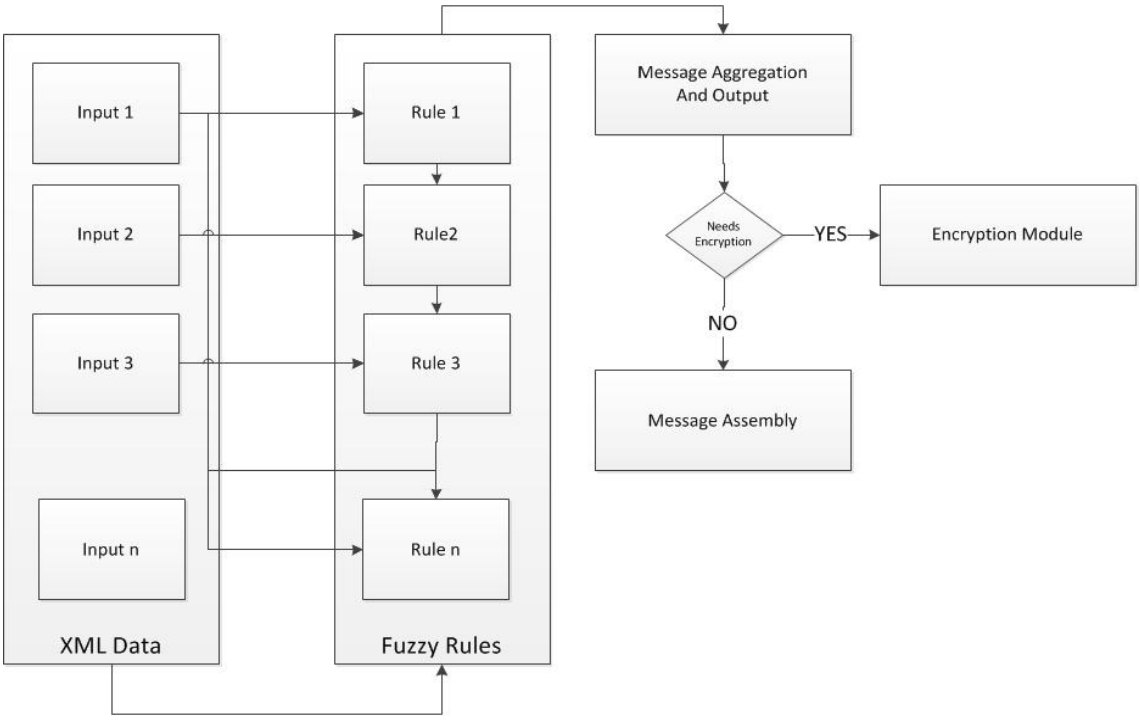

Figure 5.2: Process for XML data fuzzy classification

Step 1: This is the process of generating membership values for a fuzzy variable using membership functions. The first step is to take the crisp inputs from the 10 characteristics which stamp the importance level and determine the degree to which these inputs belong to each appropriate fuzzy set.

Step 2: The fuzzified inputs are applied to the antecedents of the fuzzy rules. Since the fuzzy rule has multiple antecedents, the fuzzy operator (AND or OR) is used to obtain a single number that represents the result of the antecedent evaluation.

Step 3: This is the process of unification of the outputs of all the rules. In other words, we are combining the membership functions of all the rules' consequents previously scaled into single fuzzy sets (output). Thus, input of the aggregation process is the list of scaled consequent membership functions, and the output is one fuzzy set for each output variable.

Step 4: This is the last step in the fuzzy inference process, where a fuzzy output of a fuzzy inference system is transformed into a crisp output. The input for the defuzzification process is the aggregate output fuzzy set where the output is a number. Centroid technique (Cox, 2001a) has been used to complete this step.

The encryption process starts just after the fuzzy classification phase, Figure 5.3 illustrate the encryption process deployment which consists of three steps.

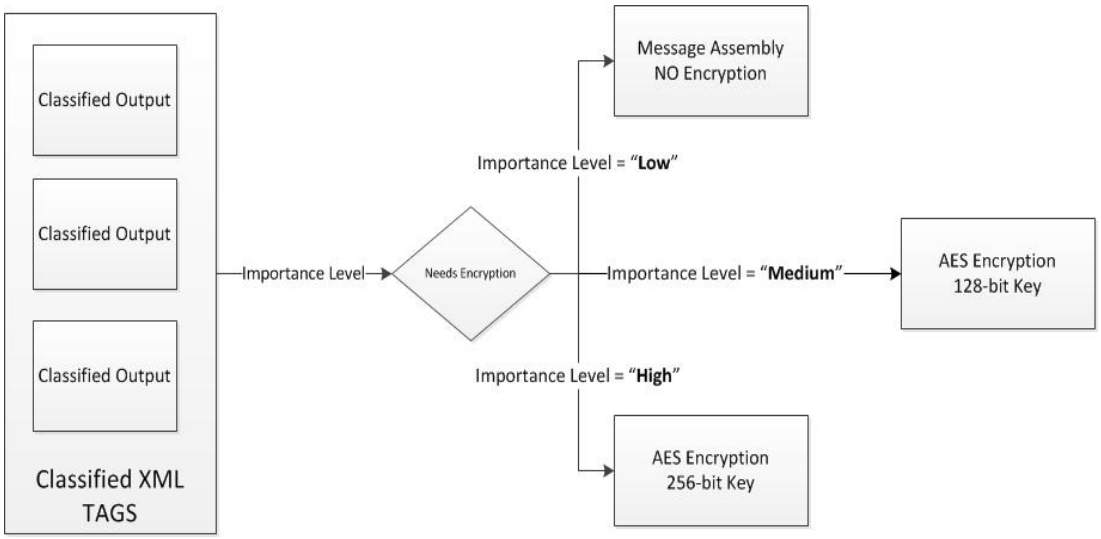

Figure 5.3: Process for XML data encryption

Step 1: This is the process of reading the value of the crisp output. Crisp output is the XML tag named "Importance Level" classified in the previous fuzzy classification process; output has one of the values (Low, Medium, and High).

Step 2: This process involves performing AES encryption on the coming XML Tags has an "Importance Level" value of "Medium" or "High". Values set to "Medium" will be encrypted using AES-128 bit key, values set to "High" will be encrypted with 256-bit key.

Step 3: This process involves redirecting message content that need no further processing into message aggregator for final message assembly.

5.4 Testing Strategy

For the purposes of testing SXMS complete model, the testing strategies are to be identified first. The next sections describe the performance testing strategy used and then the functional testing strategies.

5.4.1 Testing SXMS Behaviour

First we defined the state diagram in order to describe the behavioural activities of SXMS. State Figure 5.2 illustrates the behaviour state.

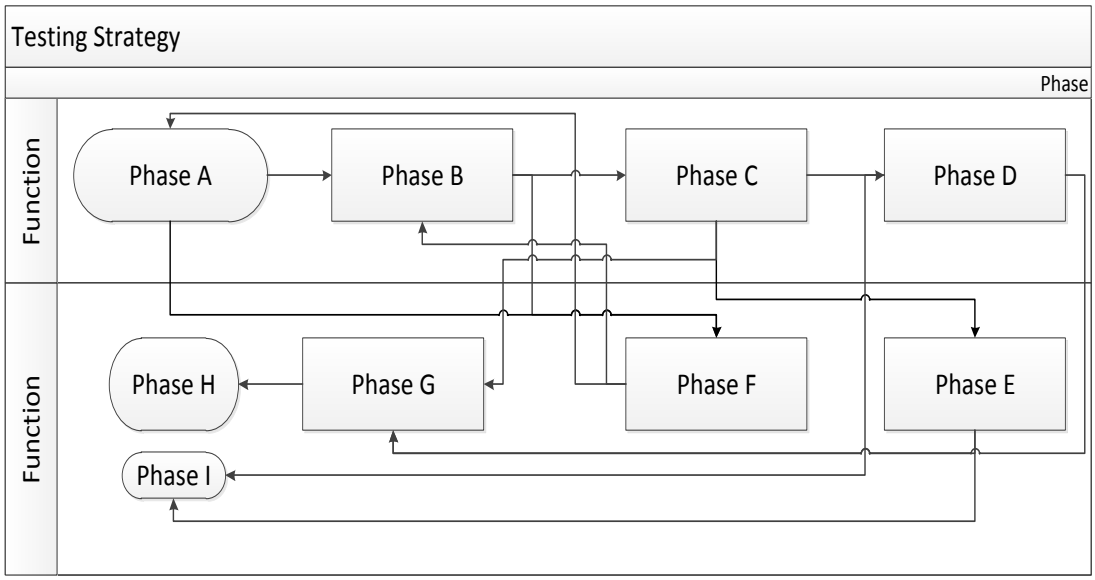

Figure 5.4: Model behaviour states

Phase-A: This phase is the GUI of the designed model. It represents the starting phase in order to deal with all the other phases. This phase has two outputs:

Out-1: to form an XML document, go to phase-B.

Out-2: to handle invalid XML document, go to phase-F.

Phase-B: This phase represents the process of parsing incoming XML documents. It has two outputs:

Out-4: to classify valid XML message, go to phase-C.

Out-5: to re-construct invalid XML message, go to phase-F.

Phase-C: This phase represents the process of classifying incoming XML messages using Mamdani fuzzy inference system and it has three outputs:

Out-6: Classified tags whereby Importance Level assigned value set to "High", go to phase-D.

Out-7: Classified tags whereby Importance Level assigned value set to "Medium", go to phase-E.

Out-8: Classified tags whereby Importance Level assigned value set to "Low", go to phase-G.

Phase-D: This phase represents the process of encrypting XML tags marked with "High" importance level using 256-bit AES encryption and it has two outputs:

Out-9: Encrypted tags using 256-bit key, go to phase-G.

Out-10: Generated private key, go to phase-I.

Phase-E: This phase represents the process of encrypting XML tags marked with "High" importance level using 128-bit AES encryption and it has two outputs:

Out-11: Encrypted tags using 128-bit key, go to phase-G.

Out-12: Generated private key, go to phase-I.

Phase-F: This phase represents the process of handling incoming error cases from different phases and it has two outputs:

Out-13: rejected XML message, go to phase-A.

Out-14: message to be modified or re-phrased, go to phase-B.

Phase-G: This phase represents the process of aggregate and assembles incoming XML tags from different phases for final composition and it has one output:

Out-15: successfully assembled message, go to phase-H.

Phase-H: This phase represents the process of message submission to final and it has one output:

Out-16: Message for final submission, submitted successfully.

Phase-I: This phase represents the process of exchanging private keys between sender and receiver and it has one output:

Out-17: Private keys for receiver.

5.4.2 Testing SXMS's Functionality

White-box and Black-box testing (IEEE, 1990) have been conducted to test the functionality of SXMS. The following describes both white-box and black-box testing steps:

5.4.2.1 White-box Testing

For our white-box testing strategy, all subroutines in our system were tested carefully by checking every statement written. Therefore, different types of XML documents were tested to guarantee all decisions, loops, nested loops, paths, and data structure have been tested. To facilitate the testing for better tracking our testing strategy, we have divided SXMS into two main sub-systems: XML Classifier, and XML Encryptor. Later we have divided each sub-system into smaller units to follow the unit testing. We have used white-box testing for the following types:

1- Unit testing (IEEE, 1990): In this testing we performed white-box testing on each unit and associated sub-units. Test cases have been written to make sure the coding is deployed correctly for further integration. We have performed this testing in an accurate way as we eliminated more than 60% of system bugs during this testing.

2- Integration Testing: in this testing we tested and evaluated the interaction between units and sub-units, interfaces between the different units are examined carefully based on test cases created earlier, some of the test cases can be used as black-box test cases as well.
3- Regression testing: we have performed a selective retesting of a set of components, sub-components, or the whole system. testing have been deployed to make sure that modifications have not caused a major effects on system behaviour, also to make sure that the system still complies with the requirements we defined (IEEE, 1990).

5.4.2.2 Black-box Testing

Although black-box testing is performed when the system is intended to be published or distributed, we performed the black-box testing to determine whether or not the system does what it supposed to do based on the functional requirements we defined. In our test we attempts to find errors and issues in external behaviour in the following categories (Pressman, 2009): (1) incorrect or missing functionality; (2) interface errors; (3) errors in data structures used by interfaces; (4)behaviour or performance errors; and (5) initialization and termination errors.

In our black-box testing a different specialist performed the testing knowing main functionality of system but without knowing all details. The following types are the black-box testing strategy types we adopted:

Integration testing: same as white-box integration testing but without knowing internal procedures and lifecycle. Testers performed their testing based on system functionality following provided work flow. Verification has been made that different units work together without issues and they are integrated into a larger code base.

Functional and system testing: we examined the high-level designs. Our functional testing involved ensuring that the functionality specified in the requirement specification works as it meant to be. System testing involves measuring the system in many different environments to ensure the program works in typical financial environments with different types of operating systems.

Acceptance testing: testing was made based on end-user expectations of the functionality. Such testing is performed to decide whether or not the system satisfies the acceptance criteria. Test cases were pre-specified and given to the testing team.

5.5 Testing Validation

In this research, we have assembled a testing framework whereby it can evaluate the encryption part of our model against the well-known W3C Encryption standard (W3C, 2001). To evaluate the effectiveness of our fuzzy XML encryption model, we have used this framework as measuring tool. We have selected data set from one of the electronic channels, which is the internet banking service in Jordan Ahli Bank (one of the leading banks in Jordan). The main goal of our testing against W3C Encryption standard is to prove the effectiveness of using SXMS model as a secure XML management tool for financial messages. Table 5.2 illustrates the factors we use in our testing and evaluation.

Table 5.2: Testing Factors

Module	Testing Factors
SXMS – Fuzzy Classification Model	• Efficiency classifying XML messaging • Functionality test
SXMS – XML Encryption	• Improve encryption processing time • File size reduction for encrypted files

5.6 Testing Environment

All experiment tests are performed on a PC (Intel core i5) 2.3 GHz CPU, 6.00 GB RAM, running Windows 7 Ultimate 64-bit operating system. NetBeans IDE 7.0 is used to implement SXMS model, and Stylus Studio 2011 XML Enterprise Suite is used as an XML editor and validator.

5.7 Testing Data Preparation

We have used two sets of data in our implementation and testing: A sample of 1,000 financial XML messages and a sample of 1,500 financial XML messages representing an internet banking service, messages extracted from Jordan Ahli Bank (JAB, 2013), one of the leading banks in Jordan. Sample has been taken for a period of seven months extracted randomly; we choose Jordan Ahli Bank as a source of our sample since the information is accessible due to our working nature within the bank and concerned departments. The extracted sample contain massive amount of historical transactions reflecting our need to test the model. Sample has the transaction amount recorded, transaction time, logged IP address, account tries, and other details used to build our fuzzy classification layers.

The collected dataset of 2,500 XML messages presenting the periods: three months (starting January 2010 until March 2010) for the first sample, and four months (April 2010 until August 2010) for the second sample of data. Both samples covered the 10 fuzzy classification characteristics we defined in our model. We have collected our samples on CD's in XML format; we arranged the documents and renamed each file to reflect the sample name and date of the transactions.

TO_ACCOUNT_NO	TO_Account_Currency	Transaction_Amount	Transaction_Currency	Posting_Date	Time_On_Site	Transaction_Time	Commission	Subscription_Number	Destination_ProfileID	Destination_Account_Tri
	Account Layer_Calc	Account Layer	Account Layer		Environment	Environment			Details Layer	Details Layer
120130101155414000	01	122.22	17	2011/03/20	2:36	7:03:00	000	999999	22	3
120130106451259000	01	100	16	11/21/2010	1:26	0:34:55	000	999999	92	3
390130106155820000	01	210	01	11/21/2010	23:46	12:54:50	000	999999	71	1
120130101100553000	01	104	02	11/21/2010	21:31	10:34:35	000	999999	68	2
120130106455673000	01	300	17	11/21/2010	11:17	10:25:12	000	999999	17	2
230230101434067000	02	800	17	11/21/2010	19:53	14:59:20	000	999999	17	2
420130106457750000	01	50	02	11/21/2010	10:04	1:49:58	000	999999	94	2
540130101478629000	01	7550	03	11/21/2010	1:24	2:56:40	000	999999	73	2
120130103100539000	01	100	01	11/21/2010	5:59	22:29:53	000	999999	32	2
420130101420248000	01	1.725	02	11/21/2010	6:13	20:02:37	000	999999	100	2
340130101415880000	01	80	16	11/23/2010	23:02	22:46:20	000	999999	15	2
250130101438706000	01	303	06	11/23/2010	9:08	21:38:38	000	999999	96	2
120230101155532000	02	1455	12	11/23/2010	16:34	4:56:40	000	999999	61	2
120130103103035000	01	30	03	11/23/2010	21:42	23:00:45	000	999999	3	2
620130101429139000	01	20000	13	11/23/2010	17:43	0:28:31	000	999999	106	2
120130106455800000	01	250	06	11/23/2010	16:05	6:28:40	000	999999	102	2
390130103103755000	01	100	16	11/23/2010	17:08	13:56:44	000	999999	49	2
580130101135295000	01	245	16	11/23/2010	8:41	13:44:14	000	999999	53	2
390130103103759000	01	100	09	11/23/2010	23:59	20:06:27	000	999999	0	2

Figure 5.5: Sample data captured from the first set

Channel_ID	Service_ID	Daily_Transactions	Transaction ID	Customer_Lang	Num_Of_Accounts	Account_Sign_Flag	Account_Currency	Acount_Balance	Account_Type	FROM_ACCOUNT_NO	Transaction_Notes_CODE	TO_ACCOUNT_NO
		Environment					Account Layer_Calc		Account Layer		Details Layer	
WWW	60	01	123456	E	01	D	17	1500	001	390230101401043000	00	120130101155414000
WWW	60	01	61520	E	01	C	16	-779.736	001	120130101451259000	00	120130106451259000
WWW	60	02	61653	E	01	C	01	-16815.021	001	390130101155820000	03	390130106155820000
WWW	60	01	62559	E	01	C	02	-10518.11	001	120130101100553000	02	120130101100553000
WWW	60	02	62612	E	01	c	17	-365.44	001	120130101455673000	02	120130106455673000
WWW	60	01	62706	E	01	C	17	-24.27	001	30130101434067600	00	230230101434067000
WWW	60	02	62782	E	01	C	02	-10518.11	001	520130101457750000	02	420130106457750000
WWW	60	01	62937	E	01	C	03	-365.44	001	560130101478629000	00	540130101478629000
WWW	60	02	47307	E	01	C	01	-24.27	001	390130106100539000	00	120130103100539000
WWW	60	03	63873	E	01	C	02	-11840.74	001	420130101420248000	00	420130101420248000
WWW	60	02	72931	E	01	D	16	59298.873	003	340130106415880000	00	340130101415880000
WWW	60	01	73210	E	01	D	06	0	001	250130106438706000	00	250130101438706000
WWW	60	01	73355	E	01	D	12	0	001	120130101155532000	02	120230101155532000
WWW	60	01	73891	E	01	C	03	-11840.74	001	120130101103035000	04	120130103103035000
WWW	60	01	73925	E	01	D	13	0	001	180130101429139000	00	620130101429139000
WWW	60	01	74112	E	01	D	06	0	001	120130106455800000	02	120130106455800000
WWW	60	02	74820	E	01	C	16	-96813.765	001	390130101103755000	00	390130103103755000
WWW	60	03	75153	E	01	C	16	-94882.42	001	580130103135295000	00	580130101135295000
WWW	60	04	75812	E	01	D	09	0	001	390130101103759000	00	390130103103759000

Figure 5.6: Sample data captured from the second set

As seen in figure 5.5, and figure 5.6, the two sample sets contain large amount of data extracted from the transaction files and arranged in two separate files. Each file has number of columns presenting the raw value that is captured during the transaction. However, we only used the 10 characteristics defined earlier to be used in our fuzzy classification stage.

5.8 Fuzzy Classification and Encryption Testing

5.8.1 Testing Fuzzy Classification

We have developed two implementations of the fuzzy classification model. The first implementation is using the first set of XML transactions presenting the internet banking service. We have selected internet banking service because it uses XML as the main messaging for data exchange between back-end host and the front-end. System has been deployed as a middleware connected to the application backend. Few customizations have been placed to match XML message structure; mapping took place as well for final message fuzzy classification .

Phase 1: First implementation conducted using a sample of 1,000 records; Records have been selected randomly presenting various transaction types like money transfers, fund transfer, and wire transfer for a period of seven months. System classified sample messages into three layers depending on 10 characteristics described in chapter 4. A sample of the structure and the records of the rule base for layer 1 are shown in Table 5.3. System main structure for layer 1 is the combination of Transaction Amount, Transaction Currency, and account type. The rule base contains (3^3) which are 27 entries and the output of this rule base is one of the Importance Level attribute fuzzy sets (High, Medium, Low) for layer 1.

Table 5.3: Sample of data classified for Layer 1

Transaction Amount	Transaction Currency	Account Type	Account Layer Importance Level Rate
Non-Sensitive	Non-Sensitive	Non-Sensitive	**Low**
Non-Sensitive	Non-Sensitive	Normal	**Low**
Non-Sensitive	Sensitive	Non-Sensitive	**Low**
Normal	Normal	Sensitive	**Medium**
Normal	Normal	Normal	**Medium**
Sensitive	Non-Sensitive	Sensitive	**High**
Sensitive	Non-Sensitive	Non-Sensitive	**Low**
Sensitive	Sensitive	Non-Sensitive	**High**

Same implementation sample has been conducted, a sample of the structure and the records of the rule base for layer 2 are shown in Table 5.4. System main structure for layer 2 is the combination of Transaction Notes, Profile ID, Account Tries, and Incorrect Password Tries. The rule base contains (3^4) which are 81 entries and the output of this rule base is one of the Importance Level attribute fuzzy sets (High, Medium, Low) for layer 2.

Table 5.4: Sample of data classified for Layer 2

Transaction Notes	Profile ID	Account Tries	Incorrect Password Tries	Details Layer Importance Level
Non-Sensitive	Normal	Non-Sensitive	Normal	**Low**
Non-Sensitive	Normal	Sensitive	Sensitive	**High**
Non-Sensitive	Sensitive	Sensitive	Sensitive	**High**
Normal	Sensitive	Normal	Sensitive	**High**
Normal	Sensitive	Non-Sensitive	Sensitive	**Medium**
Normal	Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Sensitive	Non-Sensitive	Normal	Non-Sensitive	**Low**
Sensitive	Sensitive	Sensitive	Non-Sensitive	**Medium**
Sensitive	Non-Sensitive	Non-Sensitive	Non-Sensitive	**Medium**

Same implementation sample has been conducted, a sample of the structure and the records of the rule base for layer 3 are shown in Table 5.5. System main structure for layer 3 is the combination of Time on Service, Daily Transactions, and Transaction Time. The rule base contains (3^3) which are 27 entries and the output of this rule base is one of the Importance Level attribute fuzzy sets (High, Medium, Low) for layer 3.

Table 5.5: Sample of data classified for Layer 3

Time on Service	Daily Transactions	Transaction Time	Environment Layer Importance Level Rate
Non-Sensitive	Normal	Sensitive	**High**
Non-Sensitive	Non-Sensitive	Sensitive	**Medium**
Normal	Non-Sensitive	Normal	**Medium**
Normal	Sensitive	Non-Sensitive	**Medium**
Sensitive	Normal	Sensitive	**High**
Sensitive	Sensitive	Sensitive	**High**
Sensitive	Non-Sensitive	Sensitive	**High**

The aggregated surface of the rule evaluation is defuzzified using the Mamdani method to find the Centre of Gravity (COG). Centroid defuzzification technique shown in Equation (1) can be expressed as:

$$COG = \frac{\int \mu_i(x)\ x\ dx}{\int \mu_i(x)\ dx} \qquad \text{Equation (1)}$$

μi(x): Aggregated membership function.
x: Output variable.

Figure 5.7 illustrate the set of fuzzy rules extracted from layer 1 using MATLAB R2010a, rules are based on a set of IF-THEN statements describing the effect of each factor in layer 1 on overall layer importance level. Figure 5.3 shows the surface view for layer 1 as well.

After deploying the fuzzy classification methodology on the three layers using MATLAB R2010a, we then have a list of classified tags with an importance level attribute defined and assigned.

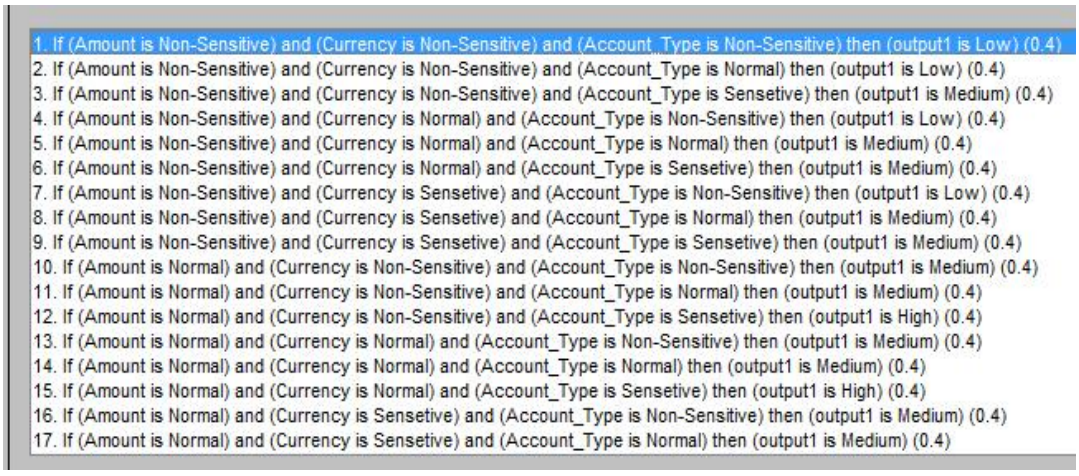

Figure 5.7: Set of fuzzy rules for layer 1

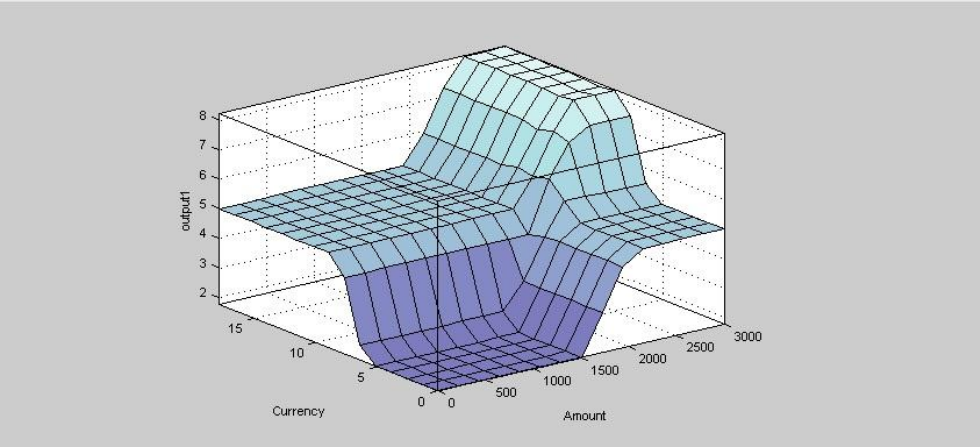

Figure 5.8: Surface view for layer 1

The fuzzy classification phase successfully processed the selected sample of 1,000 messages, The account segment layer recorded 267 out of the 1,000 sample with a "High" importance level representing 26.7% positively qualified for high level of encryption of key size of 256k. 62 out of 1,000 messages classified with a “Medium” importance level, representing 6.2% qualified for an encryption of 128k key size.

Finally 671 out of 1,000 classified with a “Low” importance level representing 67.1% of the total skipping the encryption process to be forwarded directly to message assembler.

Details segment layer achieved 401 occurrences out of the same 1,000 sample messages with a “High” importance level, representing 40.1% of the total positively qualified for high level encryption of key size of 256k. 410 out of 1,000 messages have been classified with a “Medium” importance level, representing 41% qualified for an encryption of size 128k key. The remaining 189 out of the 1,000 sample marked with “Low” importance level, representing 18.9% which require no encryption processing.

Environment segment layer recorded 250 occurrences out of the same 1,000 sample messages with a “High” importance level, representing 25% of the total positively

qualified for high level encryption of key size 256k. 421 occurrences recorded with a “Medium” importance level to be encrypted with a 128k key size. The final 329 occurrences marked with a “Low” importance level to be forwarded without any further processing. 30.6% of the occurrences average across the three layers achieved “High” importance level, 29.7% achieved “Medium” importance level, and 39.6% of the total marked “Low” importance level across the three layers.

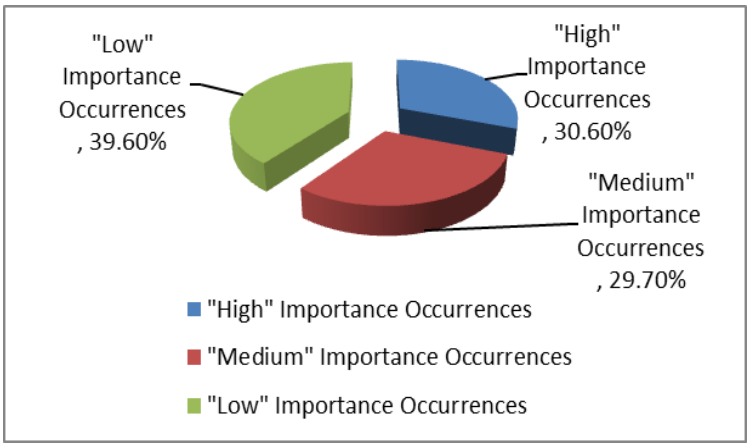

Figure 5.9: Fuzzy classification chart for sample implementation

The results clearly indicate as shown in Table 5.5 and Figure 5.10 that 26.8% of the sample 1,000 messages have a fuzzy classification level of "high" which require high encryption mechanism to secure the sensitive parts. 67% of the sample messages have been marked as "low" which have been forwarded directly without performing any encryption mechanism. The remaining 6.2% will adopt a high level of encryption but with lower key size.

Table 5.6: Fuzzy Classification table for first sample implementation

Fuzzy Classification Layer	“High” Appearances	“Medium” Appearances	Percentage (High + Medium)
Layer1 (Account)	267	62	32.9%
Layer 2 (Details)	401	410	81.1%
Layer 3 (Environment)	250	421	67.1%

As seen in table 5.6, the highest occurrences for “High” and “Medium” importance level combined is 32.9% in layer 1, which means only 32.9% of the 1,000 records sample data require an encryption processing either using 128bit key or 256bit key, leaving a 67.1% of the sample data to be forwarded directly to message assembler without the need of the encryption process. In brief, instead of performing full encryption for the whole XML message or even performing partial encryption on pre-selected parts, we were able to produce secured, optimized, and utilized messages, performing encryption only on needed parts selected using our fuzzy classification techniques based on the Mamdani method.

5.8.2 Testing XML Encryption

Assigned importance level is used as an indicator for which type of encryption is needed on corresponding node. Figure 5.10 illustrates a real XML message after fuzzy classification phase.

```
<?xml version="1.0"?>
<Transfers>
  - <Transaction ImportanceLevel="High" xmlns="http://example.org/paymentv2" Segment="Account">
        <TransactionAmount>2500</TransactionAmount>
        <Transaction_Currency Code="USD">001</Transaction_Currency>
        <Account_Type Code="001">Corporate</Account_Type>
    </Transaction>
  - <Transaction ImportanceLevel="Medium" xmlns="http://example.org/paymentv2" Segment="Details">
        <Transaction_Notes>154</Transaction_Notes>
        <Profile_ID>44</Profile_ID>
        <AccountTries>01</AccountTries>
        <PasswordTries>02</PasswordTries>
    </Transaction>
  - <Transaction ImportanceLevel="Medium" xmlns="http://example.org/paymentv2" Segment="Environment">
        <Posting_Date>2011/01/08</Posting_Date>
        <Transaction_Time>14:22:12</Transaction_Time>
        <Time_On_Service>00:25:13</Time_On_Service>
        <Daily_Transactions>2</Daily_Transactions>
        <Customer_Language>A</Customer_Language>
    </Transaction>
  - <Transaction xmlns="http://example.org/paymentv2" Segment="Additional">
        <IPAddress>128.2000.3.10</IPAddress>
        <From_Account>321456987456321457</From_Account>
        <To_Account>96325874193214587</To_Account>
    </Transaction>
</Transfers>
```

Figure 5.10: Sample XML message after fuzzy classification phase

As seen in the figure, the classified XML message has the “Importance Level” attribute with value depending on the fuzzy classification phase, “Importance

Level" attributes with a "Medium" value is entitled to have an AES Encryption using 128-bit key, "Importance Level" attributes with a "High" value is entitled to have an AES Encryption using 256-bit key, Tags with no "Importance Level" assigned will be forwarded to message assembler without performing any type of encryption. Figure 5.11 presents the XML message after deploying AES encryption on selected parts.

Figure 5.11: Sample XML message after encryption phase

As seen in Figure 5.11 only selected parts that are classified earlier are being encrypted whereby two different keys are deployed, first is AES 128-bit key on tags marked with "Medium" "Importance Level", second is AES 256-bit key on tags marked with "High" "Importance Level". Tags marked with "Low" or not marked with any value will be forwarded to message assembler without performing any type of encryption.

5.9 Chapter Summary

In this chapter, extensive tests were carried out to check the performance of our model. The model was tested against W3C XML Encryption standard. Testing took place taking into consideration element-wise encryption and full encryption. Element-wise encryption was performed and evaluated based on two cases, the first by encrypting a pre-defined list of tags within each XML message and measuring the results against our model which uses element-wise encryption on a classified list of tags, tags were selected based on a previous step which classify the XML messages deploying fuzzy classification techniques. The second case is by encrypting the whole messages and comparing the results against our element-wise based encryption model. Results showed a significant improvement in both cases presenting the superiority of our basic concept which uses on-the-fly fuzzy classification mechanism for the final goal which is encrypting necessary parts.

Chapter 6

Performance Evaluation - Implementation to Secure Financial XML Messages using Intelligent Fuzzy-Based Techniques

6.1 Introduction

Chapter 6 presents the performance evaluation, evaluation methods, and actual testing that have been performed on real data sets to properly evaluate our secure XML management model. We also present the main application classes that have been used to classify outgoing XML documents, encrypt classified XML documents, and forwarders that used to aggregate message parts. Main application code which was built using Java J2EE also presented, describes module functionality and interface. XML sample set extracted from our collected financial messages which have been used in our testing and evaluation stages. Finally, we summarize and conclude the chapter.

6.2 SXMS Performance Evaluation

For our implementation of the secure XML management system and for performance measurement, we have developed our own desktop application built based on Java technology. Our application is used to perform both fuzzy classification and encryption of XML messages. We used Java programming language (J2SE1.6) to build the main application and user interface. Application is independent and can be used on different operating systems.

6.2.1 Evaluation Method

Two parameters have been taken into account in the evaluation process: the processing time that is used to encrypt the classified messages and the size of each message after encryption process. The results are compared against the W3C Encryption standard.

- Processing Time: measure the time needed to fetch essential parts within each XML message and then encrypt needed tags using different key sizes, the measurement is based on the time needed from the starting point which is start of the XML message until the closing tag. Encryption process is performed on the whole message, selected tags, or no tags. All depending on the fuzzification stage prior to message encryption phase.
- Processing File Size: measure the file size for each XML message, size is calculated after the encryption process take place. Resulting file size from our model will be compared against the file size from the W3C Encryption standard.

6.2.2 Evaluation Preparations

Prior to the encryption phase which will be comparing SXMS model with W3C XML Encryption recommendation, we deployed the fuzzification stage whereby we classified XML message sets that are going to be used in the evaluation stages. Fuzzification

process took place to assign an importance level value to the 10 characteristics extracted from the financial XML messages. Fuzzification stage performed by deploying set of "IF-THEN" conditions, these conditions reflect the relations between the different financial features and characteristics, and also it reflects the association with each other. Relation and association with each other are used for the final importance level value. Figure 6.1 shows a sample group of IF-THEN rules used in fuzzification process that is used to assign an importance level attribute a value between (High, Medium, and Low).

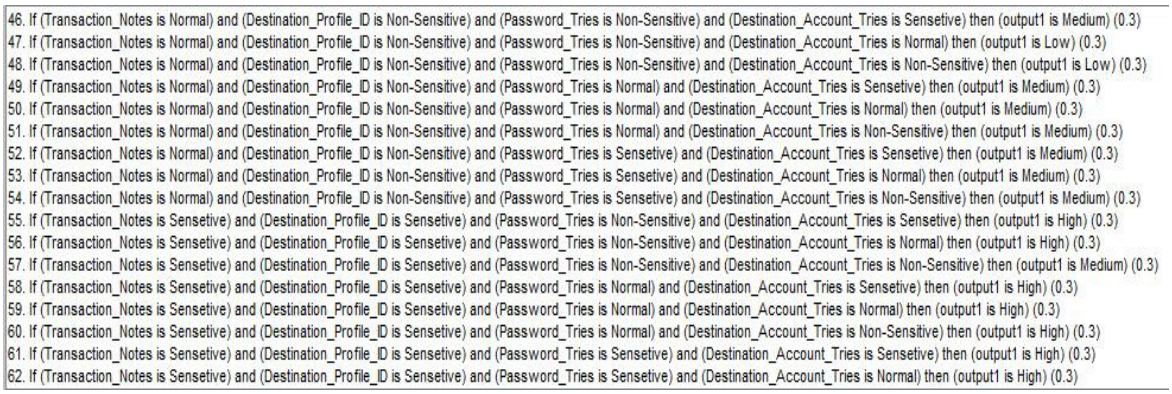
46. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Sensetive) then (output1 is Medium) (0.3)
47. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Normal) then (output1 is Low) (0.3)
48. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Low) (0.3)
49. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Sensetive) then (output1 is Medium) (0.3)
50. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3)
51. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3)
52. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Sensetive) and (Destination_Account_Tries is Sensetive) then (output1 is Medium) (0.3)
53. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Sensetive) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3)
54. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Sensetive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3)
55. If (Transaction_Notes is Sensetive) and (Destination_Profile_ID is Sensetive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Sensetive) then (output1 is High) (0.3)
56. If (Transaction_Notes is Sensetive) and (Destination_Profile_ID is Sensetive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Normal) then (output1 is High) (0.3)
57. If (Transaction_Notes is Sensetive) and (Destination_Profile_ID is Sensetive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3)
58. If (Transaction_Notes is Sensetive) and (Destination_Profile_ID is Sensetive) and (Password_Tries is Normal) and (Destination_Account_Tries is Sensetive) then (output1 is High) (0.3)
59. If (Transaction_Notes is Sensetive) and (Destination_Profile_ID is Sensetive) and (Password_Tries is Normal) and (Destination_Account_Tries is Normal) then (output1 is High) (0.3)
60. If (Transaction_Notes is Sensetive) and (Destination_Profile_ID is Sensetive) and (Password_Tries is Normal) and (Destination_Account_Tries is Non-Sensitive) then (output1 is High) (0.3)
61. If (Transaction_Notes is Sensetive) and (Destination_Profile_ID is Sensetive) and (Password_Tries is Sensetive) and (Destination_Account_Tries is Sensetive) then (output1 is High) (0.3)
62. If (Transaction_Notes is Sensetive) and (Destination_Profile_ID is Sensetive) and (Password_Tries is Sensetive) and (Destination_Account_Tries is Normal) then (output1 is High) (0.3)

Figure 6.1: Fuzzification stage (IF-THEN operators)

6.2.3 Evaluation Stages

We have performed our evaluation using two sets of XML messages; each set represent a period in which the messages were extracted. Each set has number of XML messages to test. Collected XML messages present online banking service transactions fetched from Jordan Ahli Bank, one of the leading banks in Jordan.

We have selected to deploy full and partial encryption on selected sets of XML messages, whereby we will deploy full encryption on first set of XML messages, and partial encryption on the second set of XML message. The two sets have been selected randomly taken for a period of seven months (between January 2012 until August 2012) representing financial transactions in specific. In the first set we collected 1,000 random XML messages presenting a period of three months taken between January 2012 and

March 2012. In the second set we used 1,500 XML messages presenting a period of four months taken between April 2012 and August 2012. Sample sets have been collected after taking necessary approvals and authorizations from the bank's concerned departments. Table 6.1 illustrates the two sets of XML messages in details.

Table 6.1: Stage 1 Sets detail

Set	Number of XML messages	Total Root Nodes	Total Messages Size	Period	Encryption Performed
1	1,000 messages	4,000 node	**947 KB**	3 Months Jan 12-Mar12	Full Encryption
2	1,500 messages	6,000 node	**1380 KB**	4 Months Apr12-Aug12	Partial Encryption

As seen in table 6.1, the first set has 1,000 XML messages with 4,000 root nodes and total size of 947KB. The second set has 1,500 XML messages with 6,000 root nodes and total size of 1380KB.

To ensure we are evaluating our model in a fair and comprehensive manner, we divided our evaluation into two stages. Evaluation stages are compared against W3C XML Encryption Recommendations. In each stage there are two experiments performed, each experiment presents an encryption using different key sizes. In first stage we have deployed full message encryption using W3C encryption standard with different key sizes. In the second stage we have deployed partial encryption using W3C encryption standard with different key sizes. Results from both stages are compared against our model which uses element-wise encryption and mixture of key sizes. Table 6.2 illustrates stage 1 evaluation details.

Table 6.2: Performance evaluation for stage 1

Stage	Number of XML Messages	Model	Experiment 1 Used Key Size	Experiment 2 Used Key Size
1	1,000 Messages (4,000 Nodes)	W3C (Full Encryption)	**128 bit**	**256 bit**
		SXMS (Element-Wise)	(**128** bit or **256** bit or NO Encryption)	(**128** bit or **256** bit or NO Encryption)

Stage 1: Evaluation for this stage has been conducted by performing two experiments; first experiment deployed performing full encryption using W3C XML encryption standard with a 128-bit key size deployed on the first set of 1,000 sample XML messages. SXMS uses the same sample of XML messages to deploy element-wise encryption. SXMS model uses symmetric AES encryption with mixed key values (128-bit, 256-bit), Key size used in the encryption process depends on the importance level attribute value assigned by the fuzzification stage for selected set of tags within each XML message. Second experiment has been conducted performing full encryption using W3C XML encryption standard with a 256-bit key deployed on the same 1,000 sample XML messages. SXMS uses the same sample of XML messages to deploy element-wise encryption. Later we compared results for both experiments against results from our model.

Table 6.3 illustrates time needed and resulting file size to encrypt the XML message set using our model compared against W3C XML encryption model using a key size of 128 bit encrypting each message in full.

Table 6.3: Processing time and resulting file sizes using SXMS and W3C-128

Stage 1 – Experiment 1 (Full Encryption)	**Processing Time**		**File Size**	
XML Message Set	**SXMS Model**	**W3C 128 bit**	**SXMS Model**	**W3C 128 bit**
1 XML File	0.0018 ms	0.0023 ms	1.14 KB	1.87 KB
300 XML Chunk	0.562 ms	0.702 ms	167.9 KB	263 KB
600 XML Chunk	0.873 ms	1.264 sec	342.6 KB	541.9 KB
900 XML Chunk	1.271 sec	1.825 sec	501.9 KB	799.7 KB
1,000 XML (Set 1)	**1.625** sec	**2.456** sec	**652.4 KB**	**988 KB**

We have encrypted the XML messages in chunks of 1, 300, 600, 900, and 1,000 messages. Our SXMS model processed the XML chunks with a significant

improvement in processing time compared to W3C XML encryption model which uses a 128-bit key size to encrypt the whole XML message.

SXMS uses a 128-bit key in the cases where the importance level attribute value equals to “Medium” and 256-bit key used when the importance level attribute value equals to “High”. As seen in table 6.3, the encryption process for the whole XML 1,000 messages using W3C Encryption standard with a 128-bit key size took 2.456 seconds to complete, compared to 1.625 seconds using SXMS model. The result reflects a 33.8% improvement in processing time for the 1,000 messages. Figure 6.2 illustrates the comparison between the two models and performance improvement using SXMS. Table 6.3 also illustrates files size reduction encrypting XML messages using SXMS model, table shows a significant reduction in file size, whereby the total size of the encrypted 1,000 XML messages was 988 KB using W3C model with a key size of 128-bit encrypting each XML message in full. SXMS achieved smaller sizes for the same set of 1,000 encrypted XML messages which is 652.4 KB showing a size reduction of 34% from the encrypted file size using W3C model. Such improvement can save a significant amount of space and bandwidth on large scale. Figure 6.2 illustrates the processing time needed to encrypt the sample messages in the first experiment compared to our model.

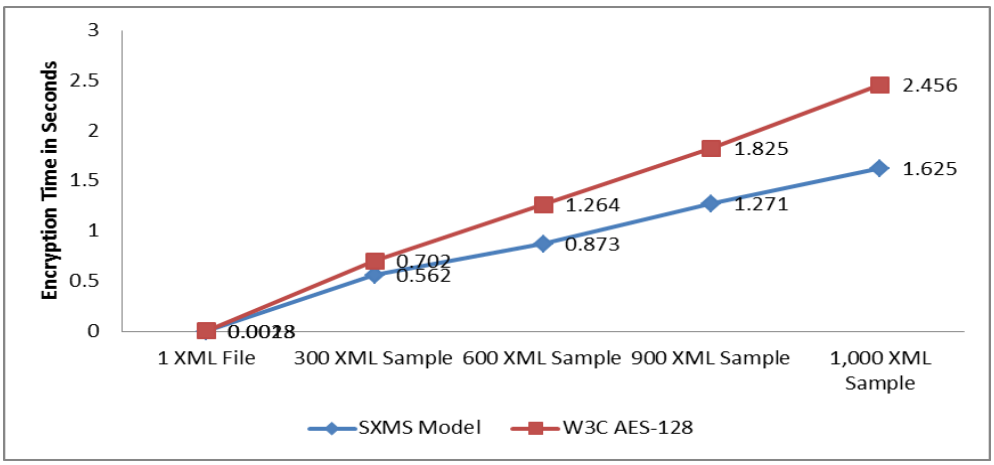

Figure 6.2: Comparison chart between SXMS and W3C model using 128-bit key

As seen in Figure 6.2, the x-axis present the number of XML messages being processed, while y-axis present the processing time encrypting XML messages in seconds.

Figure 6.3 presents file size comparison for the encrypted XML messages using SXMS and W3C XML Encryption syntax and processing model using a key size of 128-bit performing full message encryption.

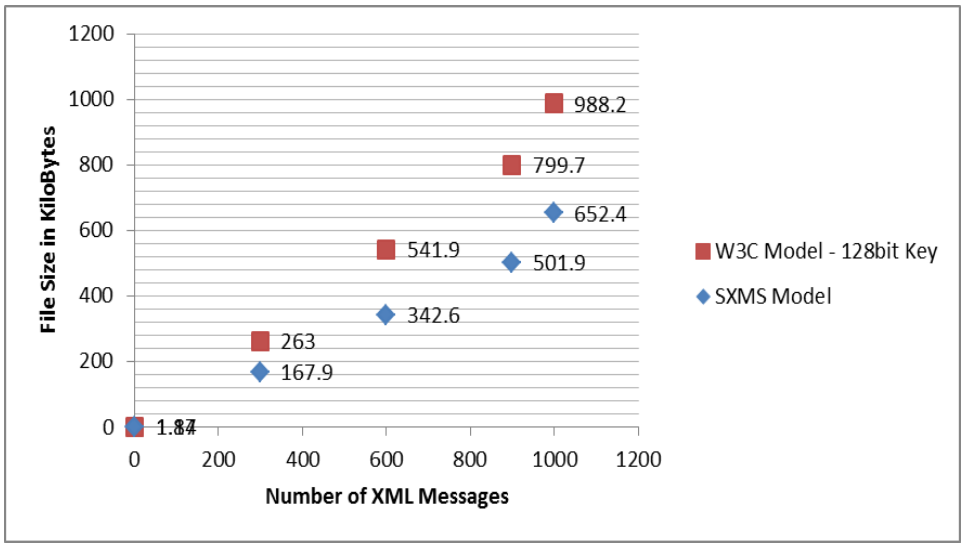

Figure 6.3: File size comparison between SXMS and W3C model using 128-bit

In the second experiment of stage 1, we deployed W3C Encryption standard to fully encrypt the same sample of 1,000 XML messages but this time using 256-bit key size. SXMS uses the same sample of XML messages to deploy element-wise encryption. SXMS model uses symmetric AES encryption with mixed key values (128-bit, 256-bit), Key size used in the encryption process depends on the importance level attribute value assigned by the fuzzification stage for selected set of tags within each XML message.

Table 6.4 represents the time needed for each model performing the encryption process on selected sample of messages.

Table 6.4: Processing time table using SXMS and W3C-256 Model (Full Encryption)

Stage 1 – Experiment 2 (Full Encryption)	**Processing Time**		**File Size**	
XML Message Set	**SXMS Model**	**W3C 256 bit**	**SXMS Model**	**W3C 256 bit**
1 XML File	0.0018 ms	0.0027 ms	1.14 KB	1.98 KB
300 XML Chunk	0.562 ms	0.811 ms	167.9 KB	283.4 KB
600 XML Chunk	0.873 ms	1.591 sec	342.6 KB	601 KB
900 XML Chunk	1.271 sec	2.137 sec	501.9 KB	864.8 KB
1,000 XML (Set 1)	**1.625** sec	**2.8** sec	**652.4 KB**	**1112 KB**

As seen in Table 6.4, the encryption process for the whole message using the W3C Encryption standard with a 256-bit key size took 2.8 seconds to complete, compared to 1.625 seconds using SXMS model. The result reflects a 41.9% improvement in processing time for the 1,000 messages.

Table 6.4 also illustrates files size reduction encrypting XML messages using SXMS model, table shows a significant reduction in file size, whereby the total size of the encrypted 1,000 XML messages was 1112 KB using W3C model with a key size of 256-bit encrypting each XML message in full. SXMS achieved smaller sizes for the same set of 1,000 encrypted XML messages which is 652.4 KB showing a size reduction of 41.3% from the encrypted file size using W3C model. Such improvement can save a significant amount of space and bandwidth on large scale. Figure 6.4 illustrates the performance comparison between SXMS model and W3C encryption standard using key size of 256-bit. Figure 6.5 presents file size comparison for the encrypted XML messages using SXMS and W3C XML Encryption syntax and processing model using a key size of 256-bit performing full message encryption.

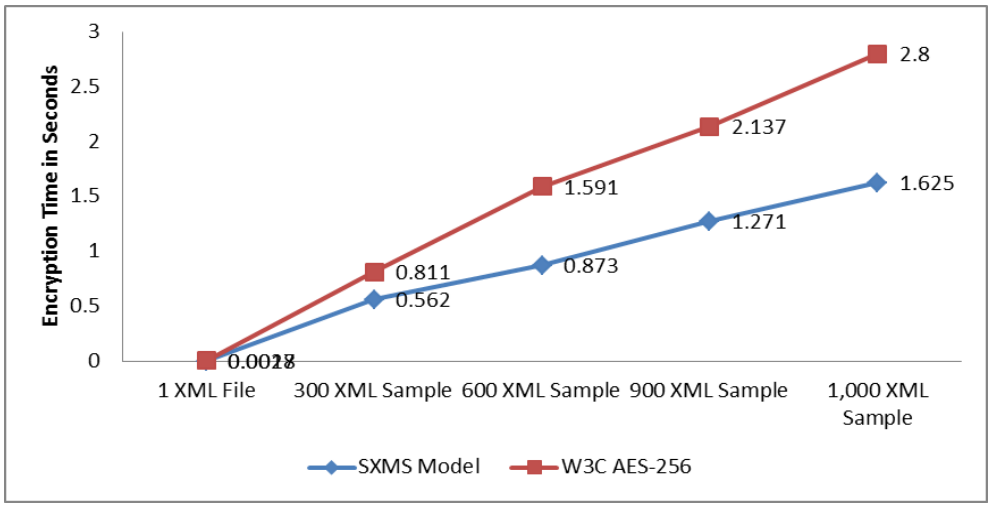

Figure 6.4: Performance comparison between SXMS and XML using 256-bit

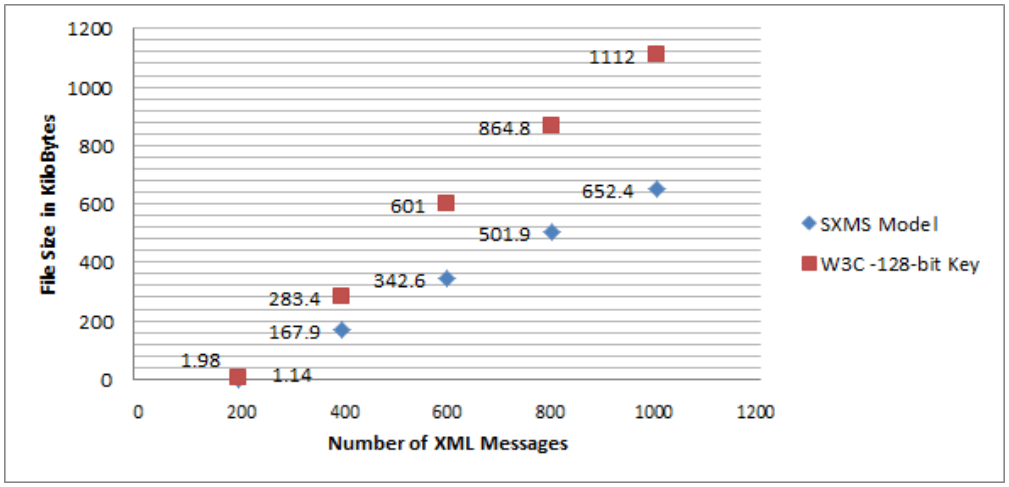

Figure 6.5: File Size comparison between SXMS and XML using 256-bit key

Finally, figures 6.6, 6.7 illustrates the final performance and file size reduction comparison between SXMS and W3C model for both experiments which uses 128-bit key and 256-bit key performing full encrypting for each XML message in the first message set. Figure presents a significant amount of performance improvement using SXMS model.

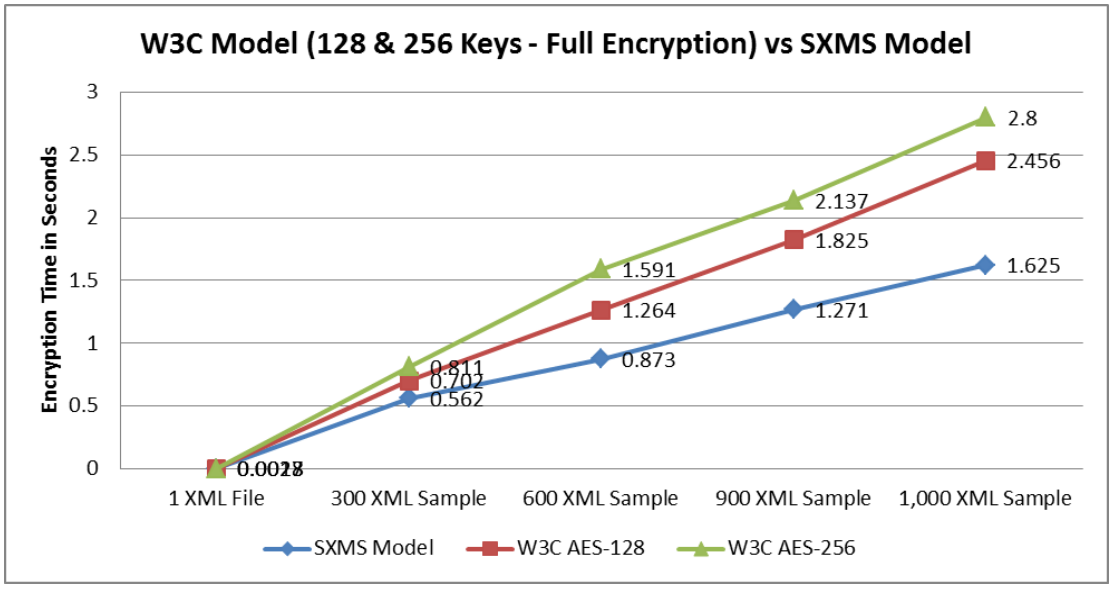

Figure 6.6: Final performance comparison between SXMS and W3C models

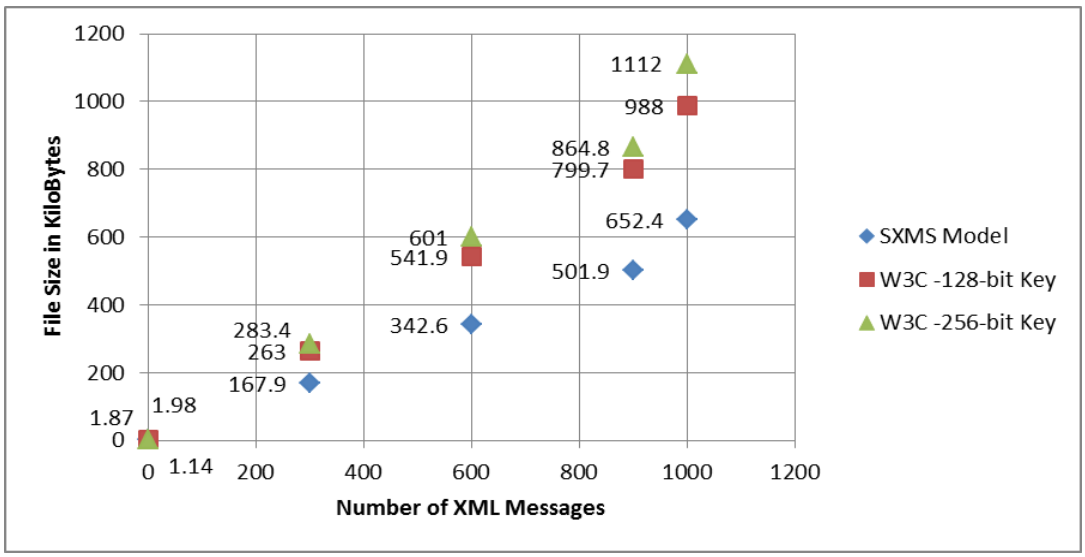

Figure 6.7: Final file size comparison between SXMS and W3C models

Table 6.5: Performance evaluation for stage 2

Stage	Number of XML Messages	Model	Experiment 1 Used Key Size	Experiment 2 Used Key Size
2	1,500 Messages (6,000 Nodes)	W3C (Partial Encryption)	**128 bit**	**256 bit**
		SXMS (Element-Wise)	(**128** bit or **256** bit or NO Encryption)	(**128** bit or **256** bit or NO Encryption)

Stage 2: Evaluation for this stage has been conducted by performing two experiments; first experiment deployed performing partial encryption on a pre-defined list of tags using W3C XML encryption standard with a 128-bit key size deployed on the second set of 1,500 sample XML messages. SXMS uses the same sample of XML messages to deploy element-wise encryption. SXMS model uses symmetric AES encryption with mixed key values (128-bit, 256-bit), Key size used in the encryption process depends on the importance level attribute value assigned by the fuzzification stage for selected set of tags within each XML message. Second experiment has been conducted performing partial encryption on a pre-defined list of tags using W3C XML encryption standard with a 256-bit key deployed on the same 1,500 sample XML messages. SXMS uses the same sample of XML messages to deploy element-wise encryption. Later we compared results for both experiments against results from our model.

Table 6.6 presents time needed to encrypt the sample messages using our model compared against W3C XML encryption model using a key size of 128 bit to encrypt part of the message for the whole set.

Table 6.6: Processing time table using SXMS and W3C-128 Model (Partial Encryption)

Stage 2 – Experiment 1 (Partial Encryption)	**Processing Time**		**File Size**	
XML Message Set	**SXMS Model**	**W3C 128 bit**	**SXMS Model**	**W3C 128 bit**
1 XML File	0.0018 ms	0.0019 ms	1.14 KB	1.61 KB
300 XML Chunk	0.562 ms	0.578 ms	167.9 KB	244 KB
600 XML Chunk	0.873 ms	0.984 sec	342.6 KB	510.2 KB
900 XML Chunk	1.271 sec	1.422 sec	501.9 KB	740.7 KB
1,500 XML (Set 2)	**1.963** sec	**2.218** sec	**810.1 KB**	**1203.6 KB**

As seen in Table 6.6, the encryption process for part of the message using the W3C Encryption standard with a 128-bit key size took 2.218 seconds to complete, compared to 1.963 seconds using SXMS model. The result reflects an 11.4% improvement in processing time for the 1,500 messages.

Table 6.6 also illustrates files size reduction encrypting XML messages using SXMS model, table shows a significant reduction in file size, whereby the total size of the encrypted 1,500 XML messages was 1203.6 KB using W3C model with a key size of 128-bit encrypting each XML message partially. SXMS achieved smaller sizes for the same set of 1,500 encrypted XML messages which is 810.1 KB showing a size reduction of 32.6% from the encrypted file size using W3C model. Such improvement can save a significant amount of space and bandwidth on large scale. Figure 6.8 illustrates the comparison between SXMS model and W3C encryption standard using key size of 128-bit.

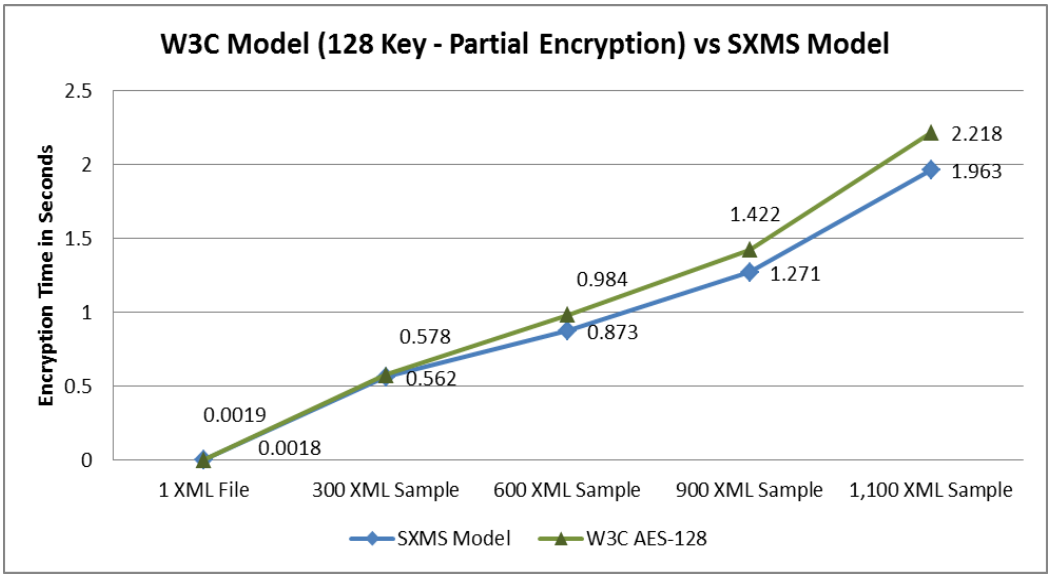

Figure 6.8: Performance comparison between SXMS and W3C Standard using AES-128 Key

Figure 6.9 presents file size comparison for the encrypted XML messages using SXMS and W3C XML Encryption syntax and processing model using a key size of 128-bit performing partial message encryption.

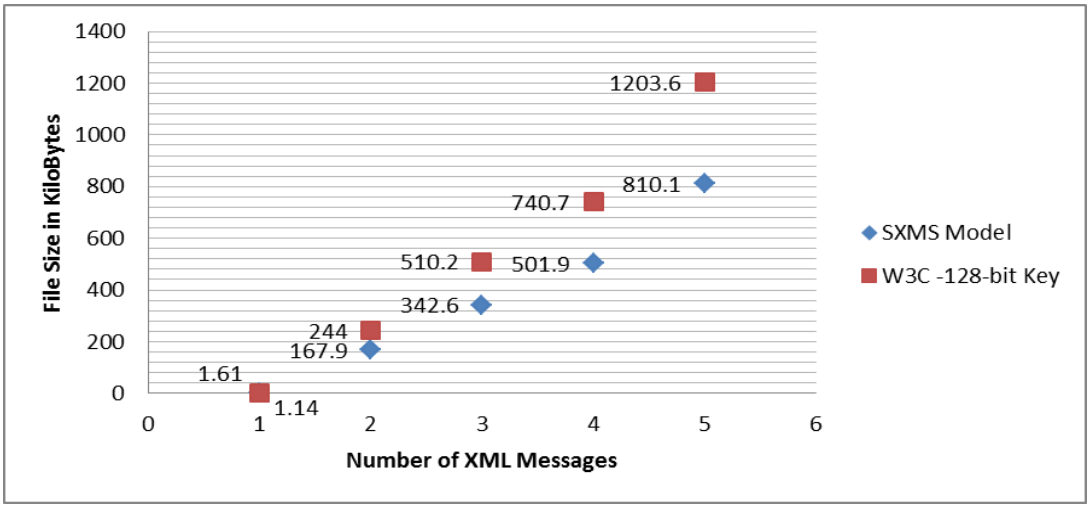

Figure 6.9: File size comparison between SXMS and W3C Standard using AES-128 Key

In the second experiment of stage 2, we deployed W3C Encryption standard to partially encrypt the XML messages to same sample of 1,500 XML messages but this time using 256-bit key size. SXMS uses the same sample of XML messages to deploy element-wise encryption. SXMS model uses symmetric AES encryption with mixed key values (128-bit, 256-bit), Key size used in the encryption process depends on the importance level attribute value assigned by the fuzzification stage for selected set of tags within each XML message. Table 6.4 represents the time needed for each model performing the encryption process on selected sample of messages.

Table 6.7: Processing time table using SXMS and W3C-256 Model (Partial)

Stage 2 – Experiment 2 (Partial Encryption)	**Processing Time**		**File Size**	
XML Message Set	**SXMS Model**	**W3C 256 bit**	**SXMS Model**	**W3C 256 bit**
1 XML File	0.0018 ms	0.0021 ms	1.14 KB	1.72 KB
300 XML Chunk	0.562 ms	0.687 ms	167.9 KB	269 KB
600 XML Chunk	0.873 sec	1.42 sec	342.6 KB	588.4 KB
900 XML Chunk	1.271 sec	2.026 sec	501.9 KB	813.9 KB
1,500 XML (Set 2)	**1.963** sec	**2.899** sec	**810.1 KB**	**1399.6 KB**

As seen in Table 6.7, the encryption process for part of the message using the W3C Encryption standard with a 256-bit key size took 2.899 seconds to complete, compared to 1.963 seconds using SXMS model. The result reflects a 32.2% improvement in processing time for the 1,500 messages. Table 6.7 also illustrates files size reduction encrypting XML messages using SXMS model, table shows a significant reduction in file size, whereby the total size of the encrypted 1,500 XML messages was 1399.6 KB using W3C model with a key size of 256-bit encrypting parts of the XML message. SXMS achieved smaller sizes for the same set of 1,500 encrypted XML messages which is 810.1 KB showing a size reduction of 42.1% from the encrypted file size using W3C model. Such improvement can save a significant amount of space and bandwidth on large scale. Figure 6.10 illustrates the comparison between SXMS model and W3C encryption standard using key size of 256-bit encrypting parts of the XML message for the second sample set.

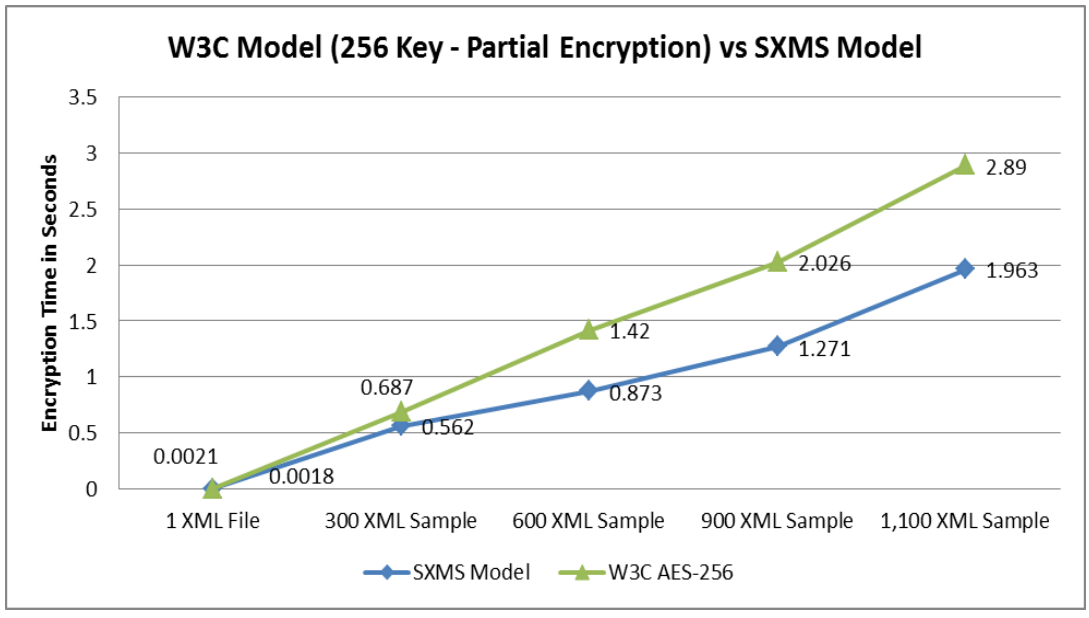

Figure 6.10: comparison between SXMS and W3C Standard using AES-256 Key

Figure 6.11 presents file size comparison for the encrypted XML messages using SXMS and W3C XML Encryption syntax and processing model using a key size of 256-bit performing partial message encryption.

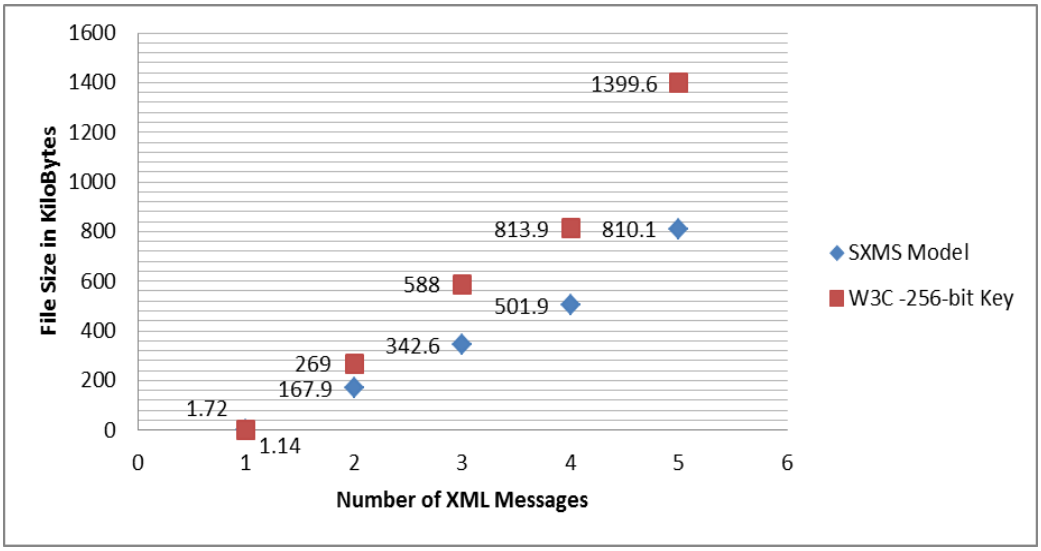

Figure 6.11: File size comparison between SXMS and W3C model using 256-bit key

Finally, figures 6.12, 6.13 illustrate performance improvements and file size reduction comparison between SXMS model and W3C model for both experiments in stage 2 showing a significant amount of performance improvement and size reduction on a large scale using SXMS model.

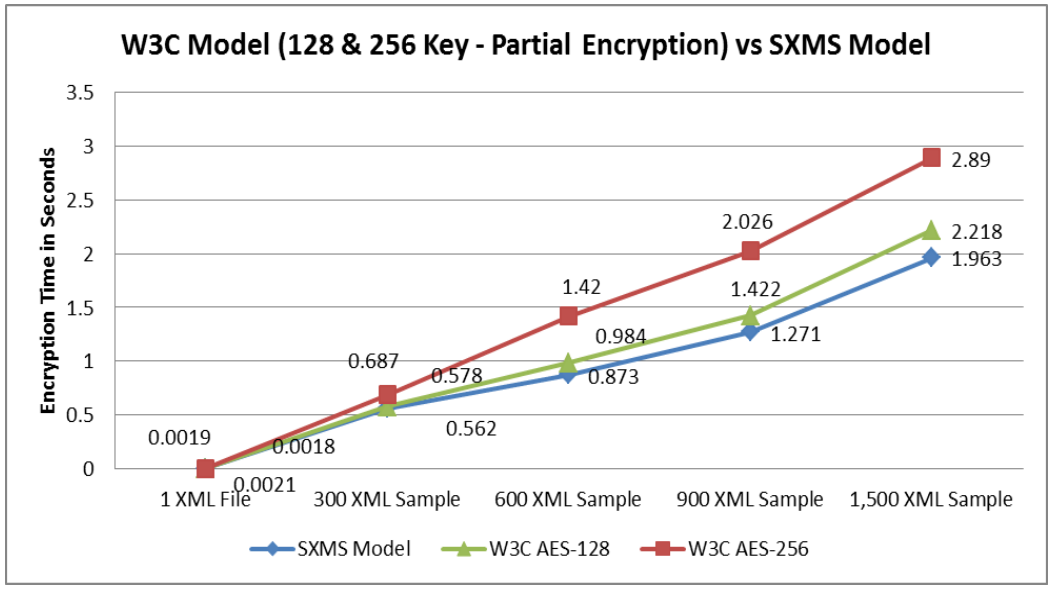

Figure 6.12: comparison between SXMS and W3C Standard using different keys

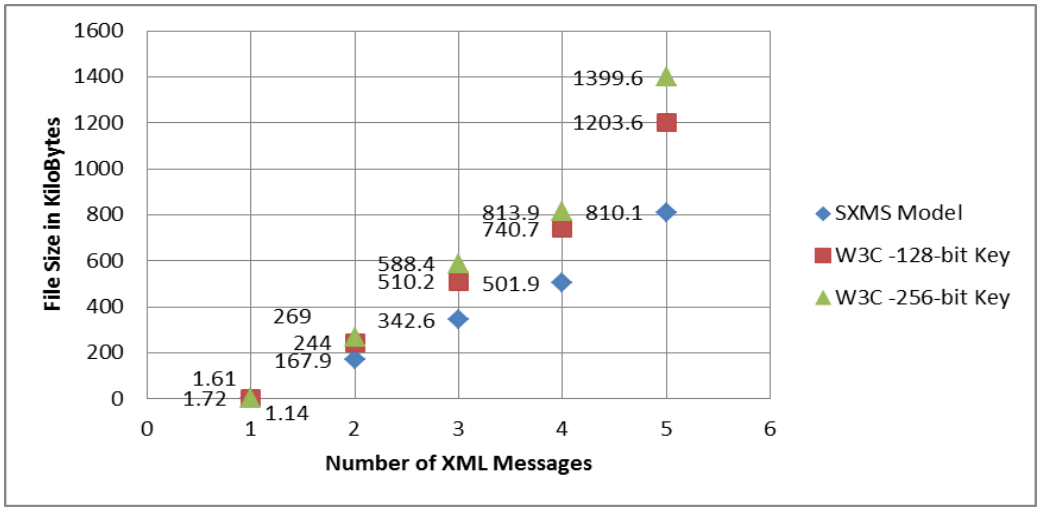

Figure 6.13: file size comparison between SXMS and W3C Standard using different keys

We have presented evidence that implementing element-wise encryption based on the fuzzy classification techniques to encrypt only necessary parts within XML messages can improve performance of the encryption process, and can reduce encrypted file sizes which eventually will save space and bandwidth on a large scale.

6.3 Screenshots, Source Codes, and Pseudo Code Examples

6.3.1 Screenshot Examples

Figure 6.14 and Figure 6.15 demonstrates screenshots of our application that is used in our testing and evaluation. Our intelligent model checked all extracted 10 characteristics within each XML message. Then using the fuzzification approach adopted by our main application, for the final importance level assigned for each tag, we have associated and classified all patterns with each other. Then we used our element-wise encryption embedded within the application to perform encryption using mixed keys depending on importance level value assigned by the fuzzification stage. The usage of AES encryption with a specific key value depends on the importance level assigned.

Whereby, tags assigned the value “High” will be encrypted and their descendants using AES encryption with a key value of 256-bit, tags assigned with “Medium” will be encrypted using a key value of 128-bit. Finally tags with “Low” importance level will be forwarded to message assembly without any encryption performed.

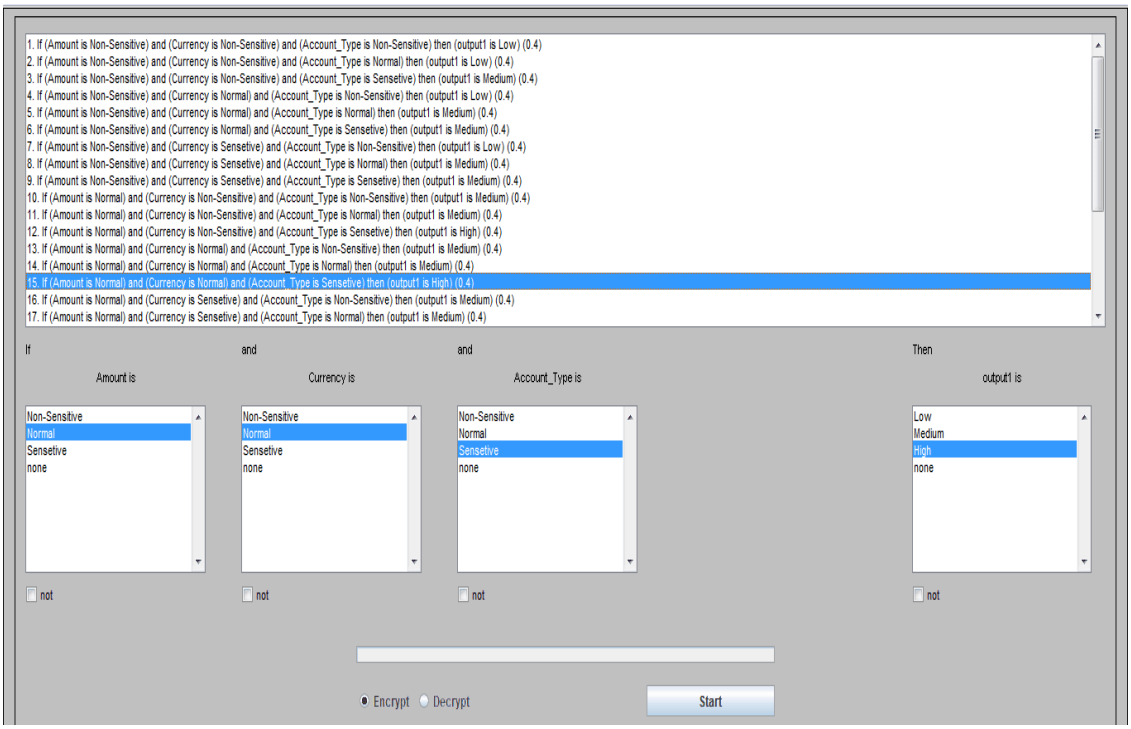

Figure 6.14: SXMS Main application interface

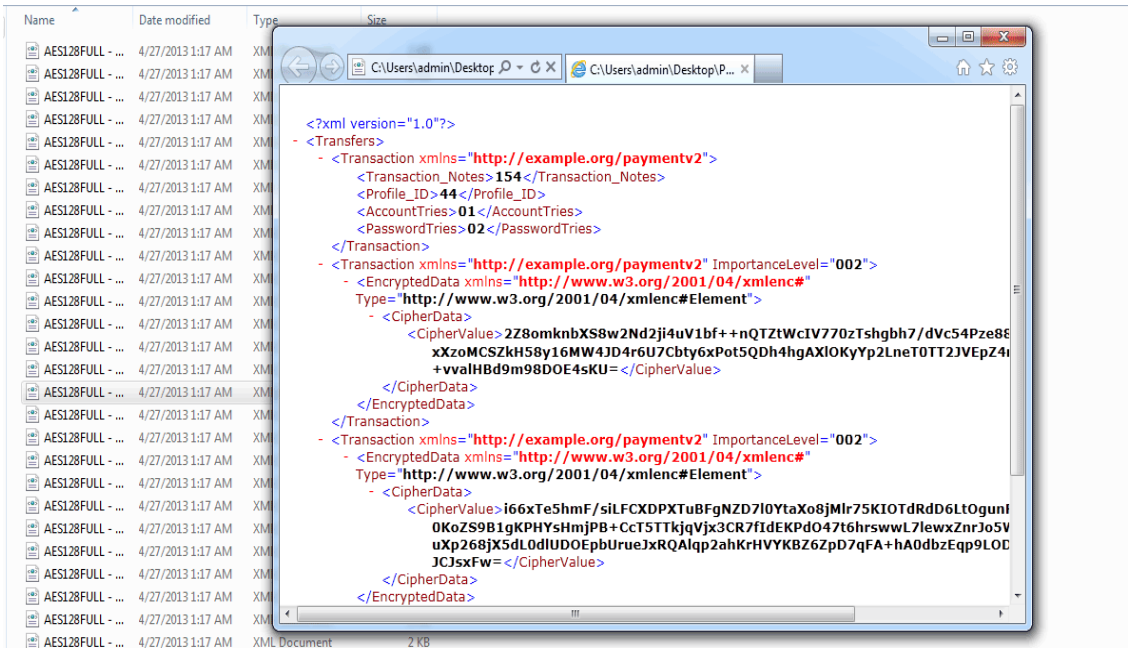

Figure 6.15: Main application result files after encryption process

6.3.2 Source Codes Examples

In this section we represent some source code examples used in our application.

- Main application source code (Java):

```
    new Thread(new Runnable(){
        public void run(){
            File dir = new File(settings.srcDir);
            File[] xmls = dir.listFiles();
            progress.setMaximum(xmls.length);
            progress.setValue(0);
            ArrayList<Object[]> times = new ArrayList<Object[]>();  //It will store xml
name and time take for encryption or decryption.

            for(File xmlName : xmls){
                Date timeStampStart = new Date();                  //Starting time of
encryption/decryption.
                progress.setValue(progress.getValue()+1);
                String fileName = xmlName.getPath();
                String xml = "";
                try {
                    xml = getXmlString(new File(fileName));        //Getting xml string from
a xml file.
                } catch (Exception ex) {
                    javax.swing.JOptionPane.showMessageDialog(null,"Error while
opening input xml file!!");
                    return;
                }
                String tagName = settings.tagName;
                String attName = settings.priorityAttribute;

                ArrayList<Data> datas = Data.parseXml(xml, tagName, attName);
//Getting list of data which need to be encrytped or decrypted.
                for(Data data : datas){
                    //Checking for key code.
                    int keycode = 128;
                    if(data.priority.equals(settings.aes256)) keycode = 256;
                    else if(data.priority.equals(settings.aes128))keycode = 128;
                    else {
                        //If key code does not match it will simply ignore it.
                        //javax.swing.JOptionPane.showMessageDialog(this, "Key code
does not match. Skipping "+data.startIndex+" To "+data.endIndex);
                        data.data2=data.data;
                        continue;
                    }

                    if(radioEncrypt.isSelected()){
                        byte[] cipherBytes=null;
                        try {
                            cipherBytes = AES.encrypt(data.data, settings.password,
keycode);       //Getting Encrypted data.
                        } catch (Exception ex) {
                            javax.swing.JOptionPane.showMessageDialog(null,"Error at the
time of encryption"+ex.toString());
                            continue;
                        }
                        data.data2= "\r\n<EncryptedData
Type='http://www.w3.org/2001/04/xmlenc#Element'
xmlns='http://www.w3.org/2001/04/xmlenc#'> \r\n" +
                                "<CipherData>\r\n<CipherValue>" + new
sun.misc.BASE64Encoder().encode(cipherBytes) +
                                "</CipherValue>\r\n</CipherData>\r\n</EncryptedData>\r\n";
                    }else{
                        try {
```

```
                        //Getting decrypted data.
                        byte[] decoded =AES.decrypt(new
sun.misc.BASE64Decoder().decodeBuffer(data.getCipherData()), settings.password,
keycode);
                        data.data2 = new String(decoded);
                    } catch (Exception ex) {
                        javax.swing.JOptionPane.showMessageDialog(null,"Error at the
time of decryption"+ex.toString());
                        return;
                    }
                }
            }

            int startIndex = 0;
            String newXml = "";
            //Recreating the xml after encryption or decryption.
            for(Data data : datas){
                newXml += xml.substring(startIndex, data.startIndex);
                newXml += data.data2;
                startIndex=data.endIndex;
            }
            newXml += xml.substring(startIndex);
            //Saving the new file to the destination folder.
            try {
                String dstFile = settings.dstDir+"\\"+xmlName.getName();
                PrintWriter pw = new PrintWriter(dstFile);
                pw.print(newXml);
                pw.flush();
                pw.close();
            } catch (FileNotFoundException ex) {
                javax.swing.JOptionPane.showMessageDialog(null,"Error when writing
final xml!!\n"+ex.toString());
            }

            //Storing the orginal file into archive folder.
            String archName = settings.arcDir+"\\"+ xmlName.getName();
            while(!xmlName.renameTo(new File(archName))){
                archName=archName.substring(0, archName.length()-4)+"_"+".xml";
            }
            //Processing ending time.
            Date timeStampEnd = new Date();
            long timeTook = timeStampEnd.getTime()-timeStampStart.getTime();
//Calculating time took for the process.
            times.add(new Object[]{xmlName.getName(),timeTook});

        }
        javax.swing.JOptionPane.showMessageDialog(null,"Done");

        //Showing time took for each file and total time.
        if(times.size()>0){
            int n = JOptionPane.showConfirmDialog(null, "Do you like to save time
stat?","AESEncrypt",JOptionPane.YES_NO_OPTION);
            if(n==JOptionPane.YES_OPTION){
                final JFileChooser fc = new JFileChooser();
                int returnVal = fc.showSaveDialog(null);
                if (returnVal == JFileChooser.APPROVE_OPTION) {
                    File file = fc.getSelectedFile();
                    try {
                        PrintWriter pw = new PrintWriter(file.getPath());
                        pw.println("FILE NAME\t\t\t\tTIME TOOK\r\n---------\t\t\t\t---------
\r\n");
                        long total=0;
                        for(Object[] obj: times){
                            String fileName = (String)obj[0];
                            long time = (Long)obj[1];
                            total += time;
                            pw.println(fileName+"\t\t\t\t"+time);
```

```
                        }
                        pw.println("---------\t\t\t\t---------\r\nTotal Time\t\t\t\t"+total+" or
"+(total/1000)+"."+(total%1000)+" Sec");
                        pw.flush();
                        pw.close();
                        javax.swing.JOptionPane.showMessageDialog(null,"Done");
                    } catch (FileNotFoundException ex) {}
                }
            }
        }

        btnContinue.setEnabled(true);
        radioEncrypt.setEnabled(true);
        radioDecrypt.setEnabled(true);
    }
}).start();
}
```

In the above code, we started by initializing the components that will be used in our process, at first we initialize the counter to check-up the processing speed needed for the whole process. Then reading XML files form incoming XML documents by getXMLString() function. Then we execute the parsing step to know which parts to be encrypted by calling Data.parseXml() function, later we define which key value to be used upon reading key string value. After deploying the encryption process for selected parts, we read the encrypted values by executing AES.encrypt(data.data, settings.password, keycode); function. Final stage is by assembling the message (with encrypted parts and forwarded parts) which requires file re-creation. Encryption time is calculated by initiating the variable ***timeStampStart*** of Date() type and running cross the application to start the counter, later another variable initiated at end of the encryption process called ***timeStampEnd*** which reads the time at the finishing stage.

- Parse XML Code:

```
public String getXmlString(File xml) throws Exception{
    String fileContent = "";
    try {
        BufferedReader reader = new java.io.BufferedReader(new
java.io.FileReader(xml.getPath()));
        String row = reader.readLine();
        while(row!=null){
            fileContent+=row+"\r\n";
            row = reader.readLine();
        }
```

```
            reader.close();
        } catch (IOException ex) {
            throw ex;
        }
        return fileContent;
    }
```

In the above class, we parse XML messages by using XPath function. getXMLString() is used to read the file by getting full file path and then read it line by line. Function returns file content to be used later in the encryption process.

- Read Encrypted XML Data

```
public String getCipherData(){
        String retVal="";
        Pattern p = Pattern.compile("<([\\s]*)CipherValue([^>]*)>([\\w\\W]*?)<([\\s]*)/([\\s]*)CipherValue([\\s]*)>");
        Matcher m = p.matcher(data);
        while(m.find()){
            retVal = m.group();
            int s = retVal.indexOf(">")+1;
            int e = retVal.lastIndexOf("<");
            retVal=retVal.substring(s, e).trim();
        }

        return retVal;
    }
}
```

In the above class, we read encrypted data by using GetCipherData() function which gets the base64-encoded data representing the encrypted form of the plaintext data.

- AES Encryption Class (Encryption):

```
public class AES {

    /**
     * Encrypt a string using AES Encryption.
     *
     * @param  data  String data to be encrypted.
     * @param  pass  Password for symmetric encryption.
     * @param  encryptSize  128 or 256 .
     */
    public static byte[] encrypt(String data, String pass, int encryptSize) throws Exception{
        try{
            byte[] keyBytes = getPassBytes(pass, encryptSize);
            byte[] input = data.getBytes();
            SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
            Cipher cipher = Cipher.getInstance("AES");
            cipher.init(Cipher.ENCRYPT_MODE, key);
```

```
            byte[] cipherText = cipher.doFinal(input);

            return cipherText;
        }catch (Exception ex) {
            throw ex;
        }
    }
```

In the AES encryption class, the cipher object is initialized. The initialization is done in cipher.init (Cipher.ENCRYPT_MODE, key); method. The first parameter determines the operation mode of the cipher. As we want to encrypt a file, we use the ENCRYPT_MODE. The second parameter is the secret key which should be used for encryption.

- **AES Decryption Class:**

```
public static byte[] decrypt(byte[] data, String pass, int encryptSize) throws Exception{
    try{
        byte[] keyBytes = getPassBytes(pass, encryptSize);
        byte[] input = data;//data.getBytes();
        SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.DECRYPT_MODE, key);
        byte[] cipherText = cipher.doFinal(input);

        return cipherText;
    }catch (Exception ex) {
        throw ex;
    }
}
```

In the AES decryption class, we start the decryption process based on the incoming message. Function calls the data string, encryption size and assign read ciphered data to a variable. Then we use the same cipher as for encryption, but we initialize it for decryption with the previously generated secret key.

6.3.3 Pseudo Codes Examples

Below are some important pseudo examples for extracting importance level value for the fuzzy classification phase in our system implementation.

- Read XML Tag “Importance Level”:

```
Value XML_Tag_Value = “Importance Level”
Number_Of_Tags;
Get Parsed_XML_Message;
Get XML_Tag_Value;

While count <= Number_Of_Tags
Check XML_Tag_Value
      If XML_Tag_Value = “High”
       Encryption_Algorithm = “AES”
       Encryption_Key = “256”
Do_Encryption_Function1 for the XML Tag: XML_Tag_Value AND Childs
      If XML_Tag_Value = “Medium”
       Encryption_Algorithm = “AES”
       Encryption_Key = “128”
Do_Encryption_Function2 for the XML Tag: XML_Tag_Value AND Childs
      If XML_Tag_Value = “Low” OR XML_Tag_Value = “”
       Encryption_Algorithm = “”
       Encryption_Key = “”
Do_Message_Forward_Function for the XML Tag: XML_Tag_Value AND Childs
```

6.4 Chapter Summary

In this chapter, extensive tests were carried out to check the performance of SXMS model and file size reduction. The basic idea of testing is to measure the encryption processing time and resulting file sizes. SXMS model was tested against W3C XML Encryption standard. Testing took place taking into consideration element-wise encryption and full encryption. Element-wise encryption was performed and evaluated based on two cases, the first by encrypting a pre-defined list of tags within each XML message and measuring the results against SXMS model which uses element-wise encryption on a fuzzified list of tags, tags were selected based on a previous step which classify XML messages deploying fuzzification techniques. The second case performed by encrypting the whole messages and comparing the results against our element-wise based encryption model. Results showed a significant improvement in both cases presenting the superiority of our basic concept which uses on-the-fly fuzzy classification mechanism encrypting only critical information within XML messages.

The other achievement is the encrypted file size reduction, whereby deploying SXMS produced smaller file sizes post the encryption process which saves both time and bandwidth on a large scale.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we have characterized an intelligent and secure XML management system has been proposed for securing financial XML messages. System has been designed to secure the essential parts within the financial XML messages. Fuzzy logic has been used to provide an efficient technique for creating an intelligent model to fetch the important parts within each XML message, which involves selecting the important parts and assigning them an attribute value called the importance level. An element-wise encryption model is used to encrypt selected parts from the fuzzy classification phase; encryption uses different key sizes during the encryption process depending on the importance level value. Set of experiments have been deployed to analyse different transaction features and patterns, with all corresponding relations. Experiments result presented the need for deploying a secure yet efficient system to handle exchanged messages in business applications in general and in financial applications in particular. The importance is not only in securing the transactions but in securing them in an

efficient and robust way. The SXMS model covered the concept of providing an efficient yet robust security mechanism for exchanging XML messages.

Phase one of SXMS uses a fuzzy-logic-based model to detect the important parts within XML messages. The fuzzy logic model we have proposed consisted of the main four stages (fuzzification stage, rule evaluation stage, aggregation stage, and finally defuzzification stage). Financial XML message features are characterized as fuzzy variables with specific fuzzy sets. For the purpose of final calculation of the XML importance level attribute value, the fuzzy rules are processed by the operations performed by fuzzy set into the inference engine.

We showed in our experiments that securing only the needed parts within each XML message is an effective mechanism to use on a large scale. In our results, we were able to achieve a significant improvement in encryption processing speed. In our first experiment, we achieved a 33.8% improvement in processing time for our sample messages. Using same sample messages, we were able to reduce file sizes for encrypted XML messages. A significant size reduction of 34% achieved using our model compared to encrypted XML messages using W3C Encryption.

In our second experiment on the same data sample, we were able to achieve improvement in encryption processing time whereby we reached 41.9% improvement using different key size. File size reduction is achieved as well hitting 41.3% improvement in our second experiment using same data sample.

We have conducted an additional two experiments using a different data sample consist of 1,500 messages. Using the same key size our model achieved 11.4% improvement in processing time and 32.6% in file size reduction.

Using same sample set of data but with different key size, we were able to achieve another improvement in encryption processing time marking 32.2% faster to encrypt the same sample and a 42.1% file size reduction for resulted encrypted files.

A desktop application has been developed and implemented. This application has extracted all the XML message features. Verification of the extracted features has been plugged into the model to identify importance level values effectively. An element-wise encryption module has been integrated into the solution to perform element-wise encryption effectively on the selected parts. The importance level attributes within each XML message that have values of “High” or “Medium” are then processed. Different key sizes are used depending on the importance level value.

The results from our evaluation stages and testing stages illustrated that our proposed solution outperformed the existing XML security solutions. Our solution outperformed these existing models in terms of efficiency, accuracy, and the speed of importance level detection, using fuzzy classification techniques.

We have presented a comparative performance of proposed system in order to illustrate the capabilities through testing stages.

Potential contributions can be deduced from our research that could enhance overall model deployment in different niches and fields. The following are the main contributions and achievements described in detail:

1. Significant improvement in processing time performing either element-wise encryption or full encryption, an average improvement in encryption processing time of 22.6% achieved compared to W3C Encryption model using the same key size. An average 34.5% improvement also achieved using different key size.

2. Significant file size reduction achieved, an average file deduction of 33.3% achieved compared to W3C Encryption model using the same key size, and a 41.2% size reduction achieved using different key size.
3. The ten features and patterns that characterize financial XML messages were extracted in a successful way and distributed in three main layers, depending on the transaction attribute nature.
4. A resilient, secure, and intelligent XML model has been proposed. The mechanism uses fuzzy logic to process the features of XML messages. The model performs element-wise encryption on selected parts with attribute values set to "High" or "Medium".
5. A desktop application has been designed and developed to test and validate our proposed model. The testing was done to prove the reliability, feasibility, and extraction process of the mechanism. The application has been developed using Java programming language; our application successfully extracted the critical and important parts within two chunks of sample messages. The two samples reflected 1,000 and 1,500 XML messages.
6. The SXMS model was built with flexibility taken into consideration. The system was designed to handle future enhancements and modifications, so researchers would be able to make core changes either at the fuzzy classification stage or at the encryption stage. Such core changes could include changing the fuzzy classification technique and how it is used; this could be used in conjunction with data mining and classification techniques, or could even be completely replaced with well-known classifiers like decision trees (Quinlan, 1979, 1986, 1998) or Classification based on association (CBA) (Liu, Hsu, & Ma, 1998). Core changes could also include changing the encryption algorithm and used keys: the AES encryption could be

replaced with DES or triple DES encryption, if the researchers could prove this change's feasibility. The flexibility of our model leaves a wide range of space for enhancements and new contributions.

7.2 Future Work

A secure XML management system has been introduced for handling the exchanging of XML messages in a secure and efficient way. Our proposed solution is based on two main phases: the first phase involves fetching the important parts within XML messages by implementing a fuzzy based classification technique. This technique uses an intelligent fuzzy methodology based on a layered structure to collect and analyse all of the XML message features and patterns. The second phase involves adopting an element-wise encryption mechanism that uses different key sizes to encrypt the parts selected in the previous stage. The level of encryption used on the selected parts depends on the attribute values assigned in the fuzzy classification stage: if the attribute value for an importance level tag is set to "High", AES encryption with a key value of 256 bits is used. A key value of 128 bits is used in the case of a "Medium" attribute value. Finally, tags with no attribute values assigned or those set to "Low" are forwarded without deploying any type of encryption.

This kind of intelligent and supervised machine learning technique provides a large amount of room and potential for future improvements and enhancements.

Each unit in our SXMS model acts independently as a separate system. This flexible nature allows and motivates future work and enhancements. Despite the significant improvement on the processing time and file size reduction, the proposed system has some limitations in term of automating the whole process, especially rule evaluation

stage whereby we need to inject the expert knowledge manually during fuzzification stage. The following points describe the future work that could be achieved to improve system functionality and automation of certain areas:

Fuzzy classification phase: We can employ supervised machine learning for fuzzy rule generation process automation. By performing this automation we can reduce the human expert involvement with high potential to increase fuzzy classification phase performance. This could be achieved by generating classification rules using well-known classifiers. For example, we could use PRISM (Cendrowska, 1987), C4.5 Decision Tree (Quinlan, 1996), Ripper (Cohen, 1995), Classification based on association (CBA) (Liu et al., 1998), k-nearest neighbour classification (kNN) (Guo et al., 2006), support vector machines (SVM) (Brank et al., 2003), naïve bayes classification (McCallum & Nigam, 1998), neural networks (NN) (Ng et al., 1997), linear least squares fit mapping (Yang & Chute, 1993), or the vector space method (Gauch et al., 2004). These association classification rules could be shared with a fuzzy logic inference engine to provide importance level extraction in an efficient way.

Encryption phase: We could utilize a different encryption scheme; asymmetric algorithms could be deployed. We have deployed symmetric encryption due to its efficiency and processing time; it outperforms asymmetric encryption algorithms. However, we could change the symmetric encryption algorithm to something different like DES, triple DES, or blowfish encryption. Researchers will be able to test and measure the performance for any replaced encryption algorithm. Also, usage of the encryption keys could be change to reflect different key sizes for each importance level assigned. For example, we could assign an encryption key size of 192 bits instead of 256 bits for the importance level “High” value.

There is also the ability to add the concept of XML classification instead of fuzzy classification or associative classification. XML classification focuses on structure similarity rather than content. We could use the classification technique using hierarchical taxonomies proposed by (Fuhr & Weikum, 2002), or we could use the well-known classifier called XRules proposed by (Zaki & Aggarwal, 2003), which focuses on the structural classification of XML documents. We also might consider a hybrid combination of content and structural classification as proposed by (Denoyer & Gallinari, 2004), which is based on a generative Bayesian classifier.

Finally, regarding the ability to generalize the Fuzzy-based Model for Financial XML Transactions Security adoption, we can expand our work in fuzzy classification phase to enable XML message parsing without the need to create a pre-defined structure. By performing this kind of improvement, we will be able to adopt the model with ease and without the need to create a pre-defined structure for message extraction. Successful adoption of this generalization enables commercial packaging for broader use in commercial and financial areas.

References

10181-1, I.-T. R. X. I. I. (1996). Information technology — Security Frameworks in Open Systems: Overview.

10181-1, I.-T. X. I. (1996). Information technology — Security Frameworks in Open Systems: Overview.

Abiteboul, S. (1996). Querying Semi-Structured Data: Stanford InfoLab.

Abiteboul, S., Segoufin, L., & Vianu, V. (2006). Representing and querying XML with incomplete information. *ACM Trans. Database Syst., 31*(1), 208-254. doi: 10.1145/1132863.1132869

Bartel, M., Boyer, J., Fox, B., LaMacchia, B., & Simon, E. (2002). XML-Signature Syntax and Processing: W3C Recommendation.

Bartel, M., Boyer, J., Fox, B., LaMacchia, B., & Simon, E. (2008). XML signature syntax and processing (second edition).

Blair, D. (2001). Re: attribute encryption (from XML encryption mailing list).

Boyer, J. (2001). *Canonical XML Version 1.0*: RFC Editor.

BOYER, J., EASTLAKE, D. E., & REAGLE, J. (2002). Exclusive XML Canonicalization, Version 1.0. Retrieved from http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/

Boyer, J., Hughes, M., & Reagle, J. (2003). *XML-Signature XPath Filter 2.0*: RFC Editor.

Brank, J., Grobelnik, M., Milic-Frayling, N., & Mladenic, D. (2003). Training text classifiers with SVM on very few positive examples.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (2007). Extensible Markup Language (XML) 1.0 (Fourth Edition). Retrieved from

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (2008). Extensible Markup Language (XML) 1.0 (Fifth Edition). Retrieved from http://www.w3.org/TR/REC-xml/

Bridges, S. M., & Vaughn, R. B. (2001). *Fuzzy Data Mining And Genetic Algorithms Applied To Intrusion Detection.* Paper presented at the 23rd National Information Systems Security Conference.

Cantor, S., Kemp, J., Philpott, R., & Maler, E. (2005). Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) v2.0. Retrieved from

Cendrowska, J. (1987). PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, *27*(4), 349-370. doi: http://dx.doi.org/10.1016/S0020-7373(87)80003-2

Clark, J. (2001). TREX - Tree Regular Expressions for XML.

Cohen, W. W. (1995). Fast Effective Rule Induction *In Proceedings of the Twelfth International Conference on Machine Learning* (pp. 115-123): Morgan Kaufmann.

Cox, E. (2001a). Fuzzy logic and the measures of certainty in eCommerce expert systems. *PC AI, 15*(3), 16-22.

Cox, E. (2001b). Fuzzy Logic and the Measures of Certainty in eCommerce Expert Systems. *Scianta Intelligence*.

Denoyer, L., & Gallinari, P. (2004). Bayesian network model for semi-structured document classification. *Inf. Process. Manage., 40*(5), 807-827. doi: 10.1016/j.ipm.2004.04.009

Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. *Information Theory, IEEE Transactions on, 22*(6), 644-654. doi: 10.1109/tit.1976.1055638

E, S., & B, L. (2000). XML encryption strawman proposal. Retrieved from http://lists.w3.org/Archives/Public/xml-encryption/2000Aug/0001.html

Eastlake, D., & Reagle, J. (2002). XML Encryption Syntax and Processing.

Ed, S. (2000). XML Encryption: Issues Regarding Attribute Values and Referenced. Retrieved from http://www.w3.org/Encryption/2001/Minutes/0103-Boston/simon-attribute-encryption.html

Ed, S. (2001). Re: attribute encryption, schema validation, role of XSLT, scope of XML encryption document (from XML encryption mailing list).

Fallside, D. (2001). XML schema part 0: primer.W3C recommendation.

Fallside, D. C., & Walmsley, P. (2004). XML Schema Part 0: Primer Second Edition. *W3C Recommendation*.

Fuhr, N., & Weikum, G. (2002). Classification and Intelligent Search on Information in XML. *Bulletin of the IEEE Technical Committee on Data Engineering, 25*, 51-58.

Galindo, J. (2008). *Handbook of Research on Fuzzy Information Processing in Databases*: Information Science Reference - Imprint of: IGI Publishing.

Gauch, S., Madrid, J. M., Induri, S., Ravindran, D., & Chadlavada, S. (2004). KeyConcept: A Conceptual Search Engine: University of Kansas.

Gaurav, A., & Alhajj, R. (2006). *Incorporating fuzziness in XML and mapping fuzzy relational data into fuzzy XML*. Paper presented at the Proceedings of the 2006 ACM symposium on Applied computing, Dijon, France.

Geuer-Pollmann, C. (2002). *XML pool encryption*. Paper presented at the Proceedings of the 2002 ACM workshop on XML security, Fairfax, VA.

GODIK, S., & MOSES, T. (2002). XACML 1.0 - The OASIS extensible Access Control Markup Language (XACML). Retrieved from http://www.oasis-open.org/committees/xacml/
http://www.oasis-open.org/committees/xacml/repository/cs-xacml-core-01.doc

Godik, S., & Moses, T. (2003). eXtensible Access Control Markup Language (XACML) Version 1.0. Retrieved from http://www.oasis-open.org/committees/xacml/repository/

Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2006). Using kNN model for automatic text categorization. *Soft Computing, 10*(5), 423-430. doi: 10.1007/s00500-005-0503-y

Hallam-Baker, P., & Mysore, S. H. (2005). XML Key Management Specification (XKMS 2.0) Retrieved June, 2010, from http://www.w3.org/TR/xkms2/

Hégaret, P. L., Whitmer, R., & Wood, L. (2005). Document Object Model (DOM) Retrieved from http://www.w3.org/DOM/

Herrera-Viedma, E., Peis, E., Morales-del-Castillo, J. M., Alonso, S., & Anaya, K. (2007). A fuzzy linguistic model to evaluate the quality of Web sites that store XML documents. *International Journal of Approximate Reasoning, 46*(1), 226-253. doi: http://dx.doi.org/10.1016/j.ijar.2006.12.010

Herrera, F., Herrera-Viedma, E., & Verdegay, J. L. (1996). Direct approach processes in group decision making using linguistic OWA operators. *Fuzzy Sets and Systems, 79*(2), 175-190. doi: http://dx.doi.org/10.1016/0165-0114(95)00162-X

Hughes, J., Cantor, S., Hodges, J., Hirsch, F., Mishra, P., Philpott, R., & Maler, E. (2005). Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0. Retrieved from

Hunter, D., Cagle, K., Dix, C., & Cable, D. (2001). *Beginning XML 2nd edition* (2nd ed.): Wrox Press.

IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, 1-84. doi: 10.1109/ieeestd.1990.101064

Imamura, T., Dillaway, B., & Simon, E. (2002). XML Encryption Syntax and Processing.

ISO10181-1, I.-T. X. Information technology — Security Frameworks in Open Systems: Overview.

ISO/IEC9798-1. (1997). Information technology — Security techniques — Entity authentication — Part 1: General.

ISO/IEC11770. Information technology — Security techniques — Key management. Retrieved from

JAB. (2013). Jordan Ahli Bank, from http://www.ahli.com/

Jelliffe, R. (2006). Resource Directory (RDDL) for Schematron 1.5. Retrieved June 2010, from http://xml.ascc.net/schematron/

Joseph, R. (2001). XML Encryption Requirements Retrieved November, 2011, from http://www.w3.org/TR/xml-encryption-req
http://www.w3.org/Encryption/2001/Drafts/xml-encryption-req

King, S. (2003). Threats and Solutions to Web Services Security. *Network Security, 2003*(9), 8-11. doi: http://dx.doi.org/10.1016/S1353-4858(03)00907-3

Klarlund, N., Møller, A., & Schwartzbach, M. I. (2000). *DSD: A Schema Language for XML*. Paper presented at the Workshop on Formal Methods in Software Practice, Portland, Oregon. http://www.brics.dk/DSD/dsd.html

Klir, G. J., & Yuan, B. (1995). *Fuzzy sets and fuzzy logic: theory and applications*: Prentice-Hall, Inc.

Lee, J., & Fanjiang, Y.-y. (2003). Modeling imprecise requirements with XML. *Information & Software Technology, 45*(7), 445-460. doi: 10.1016/s0950-5849(03)00015-6

Liu, B., Hsu, W., & Ma, Y. (1998). Integrating Classification and Association Rule Mining (pp. 80-86).

Lukasiewicz, T., & Straccia, U. (2008). Managing uncertainty and vagueness in description logics for the Semantic Web. *Web Semant., 6*(4), 291-308. doi: 10.1016/j.websem.2008.04.001

M\oller, A. (2005, December). Document Structure Description 2.0

Ma, Z. M., & Yan, L. (2007). Fuzzy XML data modeling with the UML and relational data models. *Data Knowl. Eng., 63*(3), 972-996. doi: 10.1016/j.datak.2007.06.003

Mahant, N. (2004). Risk Assessment is Fuzzy Business—Fuzzy Logic Provides the Way to Assess Off-site Risk from Industrial Installations. Bechtel, Australia: Risk 2004.

Makoto, M. (2002). RELAX (Regular Language description for XML) Retrieved June 2010, from http://www.xml.gr.jp/relax/

Makoto, M., Walsh, N., & McRae, M. (2001). TREX and RELAX Unified as RELAX NG, a Lightweight XML Language Validation Specification.

Mamdani, E. H., & Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies, 7*(1), 1-13. doi: http://dx.doi.org/10.1016/S0020-7373(75)80002-2

Maruyama, H., & Imamura, T. (2000). Element-Wise XML Encryption Retrieved February 2008, from http://lists.w3.org/Archives/Public/xml-encryption/2000Apr/att-0005/01-xmlenc

McCallum, A., & Nigam, K. (1998). A comparison of event models for Naive Bayes text classification

Mclaughlin, B., & Edelson, J. (2006). *Java and XML Third Edition*: O'Reilly.

Ng, H. T., Goh, W. B., & Low, K. L. (1997). Feature selection, perceptron learning, and a usability case study for text categorization. *SIGIR Forum, 31*(SI), 67-73. doi: 10.1145/278459.258537

Nierman, A., & Jagadish, H. V. (2002). ProTDB: Probabilistic data in XML *In Proceedings of the 28th VLDB Conference* (pp. 646-657): Springer.

O'Neill, M. (2003). *Web Services Security* (1 ed.): McGraw-Hill Osborne Media.

OASIS, S. S. T. C. o. (2002). Oasis security services (saml) Retrieved October 2010, from http://www.oasis-open.org/committees/security/

Ortega, F. B. (2008). *Managing Vagueness in Ontologies.* PhD PhD Dissertation, University of Granada, Spain.

Petrovic, D., Roy, R., & Petrovic, R. (1999). Supply chain modelling using fuzzy sets. *International Journal of Production Economics, 59*(1–3), 443-453. doi: http://dx.doi.org/10.1016/S0925-5273(98)00109-1

Phillip M, H.-B., & Ford, W. (2001). *XML Key Management Specification (XKMS)*. http://www10.org/cdrom/posters/1129.pdf

Pressman, R. (2009). *Software Engineering: A Practitioner's Approach* (7 ed.): McGraw-Hill.

Quinlan, J. R. (1979). *Discovering rules from large collections of examples: a case study*.

Quinlan, J. R. (1986). Induction of Decision Trees. *Mach. Learn., 1*(1), 81-106. doi: 10.1023/a:1022643204877

Quinlan, J. R. (1996). Improved Use of Continuous Attributes in C4.5. *Journal of Artificial Intelligence Research, 4*, 77-90.

Quinlan, J. R. (1998). *Data Mining Tools see5 and c5*.

Ray, E. T. (2003). *Learning XML - creating self-describing data: cover schemas* (2nd ed.): O'Reilly.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM, 21*(2), 120-126. doi: 10.1145/359340.359342

Rosario, R. (2001). Secure XML An Overview of XML Encryption.

Smets, P. (1996). Imperfect Information: Imprecision and Uncertainty *Uncertainty Management in Information Systems* (pp. 225-254).

Steve, W. (2001). Re: attribute encryption (from XML encryption mailing list).

Sugeno, M. (1985). An introductory survey of fuzzy control. *Information Sciences, 36*, 59-83.

Takeshi, I., & Hiroshi, M. (2000). Specification of element-wise XML encryption. Retrieved from http://lists.w3.org/Archives/Public/xml-encryption/2000Aug/att-0005/01-xmlenc-spec.html

Tseng, C., Khamisy, W., & Vu, T. (2005). Universal fuzzy system representation with XML. *Computer Standards & Interfaces, 28*(2), 218-230. doi: http://dx.doi.org/10.1016/j.csi.2004.11.005

Turowski, K., & Weng, U. (2002). Representing and processing fuzzy information — an XML-based approach. *Knowledge-Based Systems, 15*(1–2), 67-75. doi: http://dx.doi.org/10.1016/S0950-7051(01)00122-8

Violleau, T. (2001). Java Technology and XML. Retrieved from

W3C. (1998). Extensible Markup Language Retrieved October 2009, from http://www.w3.org/TR/1998/REC-xml-19980210

W3C. (2001). XML Encryption Syntax and Processing.

W3C, BOYER, J., EASTLAKE, D. E., & REAGLE, J. (2002). Exclusive XML Canonicalization, Version 1.0 Retrieved April 2011, from http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/

Weerasinghe, D., Elmufti, K., Rajarajan, M., & Rakocevic, V. (2006, Nov. 29 2006-Dec. 1 2006). *XML Security based Access Control for Healthcare Information in Mobile Environment.* Paper presented at the Pervasive Health Conference and Workshops, 2006.

Williams, I. (2009). *Beginning XSLT and XPath: Transforming XML Documents and*
Data: Wrox Press.

Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *Systems, Man and Cybernetics, IEEE Transactions on, 18*(1), 183-190. doi: 10.1109/21.87068

Yager, R. R. (2000). Targeted e-commerce marketing using fuzzy intelligent agents. *Intelligent Systems and their Applications, IEEE, 15*(6), 42-45. doi: 10.1109/5254.895859

Yager, R. R., & Pasi, G. (2001). Product category description for web-shopping in e-commerce. *International Journal of Intelligent Systems, 16*(8), 1009-1021. doi: 10.1002/int.1046

Yang, Y., & Chute, C. G. (1993). *An application of least squares fit mapping to text information retrieval*. Paper presented at the Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval, Pittsburgh, Pennsylvania, USA.

Yang, Y., & Liu, X. (1999). *A re-examination of text categorization methods*. Paper presented at the Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, Berkeley, California, USA.

Zadeh, L. A. (1965). Fuzzy Sets. *Information Control, 8*, 338-353.

Zadeh, L. A. (1984). Making computers think like people: The term `fuzzy thinkingÂ¿ is pejorative when applied to humans, but fuzzy logic is an asset to machines in applications from expert systems to process control. *Spectrum, IEEE, 21*(8), 26-32. doi: 10.1109/mspec.1984.6370431

Zaki, M. J., & Aggarwal, C. C. (2003). *XRules: an effective structural classifier for XML data*. Paper presented at the Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, Washington, D.C.

Zhang, F., Ma, Z. M., & Yan, L. (2013). Construction of fuzzy ontologies from fuzzy XML models. *Knowledge-Based Systems, 42*(0), 20-39. doi: http://dx.doi.org/10.1016/j.knosys.2012.12.015

Appendix A

Rules for Layer 1, Layer 2, and Layer 3

The complete list of rules extracted from MATLAB is listed in Appendix A. rules represent the three layers, layer 1 which is the account segment, layer 2 which is the details segment, and layer 3 which is environment segment. There are 135 rule defined in this list.

Rules for Layer 1 (Account Segment)

There are 27 rules extracted from MATLAB fuzzy set, layer 1 represent account segment.

Fuzzy Rules:

Rule 1. If (Amount is Non-Sensitive) and (Currency is Non-Sensitive) and (Account_Type is Non-Sensitive) then (output1 is Low) (0.4)
Rule 2. If (Amount is Non-Sensitive) and (Currency is Non-Sensitive) and (Account_Type is Normal) then (output1 is Low) (0.4)
Rule 3. If (Amount is Non-Sensitive) and (Currency is Non-Sensitive) and (Account_Type is Sensitive) then (output1 is Medium) (0.4)
Rule 4. If (Amount is Non-Sensitive) and (Currency is Normal) and (Account_Type is Non-Sensitive) then (output1 is Low) (0.4)
Rule 5. If (Amount is Non-Sensitive) and (Currency is Normal) and (Account_Type is Normal) then (output1 is Medium) (0.4)
Rule 6. If (Amount is Non-Sensitive) and (Currency is Normal) and (Account_Type is Sensitive) then (output1 is Medium) (0.4)
Rule 7. If (Amount is Non-Sensitive) and (Currency is Sensitive) and (Account_Type is Non-Sensitive) then (output1 is Low) (0.4)
Rule 8. If (Amount is Non-Sensitive) and (Currency is Sensitive) and (Account_Type is Normal) then (output1 is Medium) (0.4)
Rule 9. If (Amount is Non-Sensitive) and (Currency is Sensitive) and (Account_Type is Sensitive) then (output1 is Medium) (0.4)
Rule 10. If (Amount is Normal) and (Currency is Non-Sensitive) and (Account_Type is Non-Sensitive) then (output1 is Medium) (0.4)
Rule 11. If (Amount is Normal) and (Currency is Non-Sensitive) and (Account_Type is Normal) then (output1 is Medium) (0.4)
Rule 12. If (Amount is Normal) and (Currency is Non-Sensitive) and (Account_Type is Sensitive) then (output1 is High) (0.4)
Rule 13. If (Amount is Normal) and (Currency is Normal) and (Account_Type is Non-Sensitive) then (output1 is Medium) (0.4)
Rule 14. If (Amount is Normal) and (Currency is Normal) and (Account_Type is Normal) then (output1 is Medium) (0.4)
Rule 15. If (Amount is Normal) and (Currency is Normal) and (Account_Type is Sensitive) then (output1 is High) (0.4)

Rule 16. If (Amount is Normal) and (Currency is Sensitive) and (Account_Type is Non-Sensitive) then (output1 is Medium) (0.4)
Rule 17. If (Amount is Normal) and (Currency is Sensitive) and (Account_Type is Normal) then (output1 is Medium) (0.4)
Rule 18. If (Amount is Normal) and (Currency is Sensitive) and (Account_Type is Sensitive) then (output1 is Medium) (0.4)
Rule 19. If (Amount is Sensitive) and (Currency is Non-Sensitive) and (Account_Type is Non-Sensitive) then (output1 is Medium) (0.4)
Rule 20. If (Amount is Sensitive) and (Currency is Non-Sensitive) and (Account_Type is Normal) then (output1 is Medium) (0.4)
Rule 21. If (Amount is Sensitive) and (Currency is Non-Sensitive) and (Account_Type is Sensitive) then (output1 is High) (0.4)
Rule 22. If (Amount is Sensitive) and (Currency is Normal) and (Account_Type is Non-Sensitive) then (output1 is Medium) (0.4)
Rule 23. If (Amount is Sensitive) and (Currency is Normal) and (Account_Type is Normal) then (output1 is High) (0.4)
Rule 24. If (Amount is Sensitive) and (Currency is Normal) and (Account_Type is Sensitive) then (output1 is High) (0.4)
Rule 25. If (Amount is Sensitive) and (Currency is Sensitive) and (Account_Type is Non-Sensitive) then (output1 is Medium) (0.4)
Rule 26. If (Amount is Sensitive) and (Currency is Sensitive) and (Account_Type is Normal) then (output1 is High) (0.4)
Rule 27. If (Amount is Sensitive) and (Currency is Sensitive) and (Account_Type is Sensitive) then (output1 is High) (0.4)

Number of Rules: 27

Rules for Layer 2 (Details Segment)

There are 81 rules extracted from MATLAB fuzzy set, layer 2 represent details segment.

Fuzzy Rules:

Rule 1. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is Medium) (0.3(
Rule 2. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Normal) then (output1 is Low) (0.3 (
Rule 3. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Low) (0.3 (
Rule 4. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Sensitive) then (output1 is Medium) (0.3(
Rule 5. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (
Rule 6. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 7. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is High) (0.3 (

Rule 8. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Normal) then (output1 is High) (0.3 (
Rule 9. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3(
Rule 10. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is Low) (0.3 (
Rule 11. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (
Rule 12. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Low) (0.3(
Rule 13. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Normal) and (Destination_Account_Tries is Sensitive) then (output1 is Medium) (0.3 (
Rule 14. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Normal) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (

Rule 15. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Normal) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Low) (0.3(
Rule 16. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is Medium) (0.3 (
Rule 17. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (
Rule 18. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 19. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is Low) (0.3 (
Rule 20. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Normal) then (output1 is Low) (0.3(
Rule 21. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Low) (0.3(
Rule 22. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Sensitive) then (output1 is Medium) (0.3 (
Rule 23. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (

Rule 24. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Low) (0.3(
Rule 25. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is Medium) (0.3 (
Rule 26. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (
Rule 27. If (Transaction_Notes is Non-Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Low) (0.3 (
Rule 28. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is High) (0.3(

Rule 29. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Normal) then (output1 is High) (0.3 (
Rule 30. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 31. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Sensitive) then (output1 is High) (0.3(
Rule 32. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (
Rule 33. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 34. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is High) (0.3 (

Rule 35. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Normal) then (output1 is High) (0.3(
Rule 36. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is High) (0.3(
Rule 37. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Normal) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is Medium) (0.3 (
Rule 38. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Normal) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (
Rule 39. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Normal) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 40. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Normal) and (Password_Tries is Normal) and (Destination_Account_Tries is Sensitive) then (output1 is Medium) (0.3 (
Rule 41. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Normal) and (Password_Tries is Normal) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (
Rule 42. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Normal) and (Password_Tries is Normal) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 43. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Normal) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is High) (0.3(
Rule 44. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Normal) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (
Rule 45. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Normal) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 46. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is Medium) (0.3 (
Rule 47. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Normal) then (output1 is Low) (0.3(
Rule 48. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Low) (0.3(
Rule 49. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Sensitive) then (output1 is Medium) (0.3 (
Rule 50. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (

Rule 51. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 52. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is Medium) (0.3 (

Rule 53. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (
Rule 54. If (Transaction_Notes is Normal) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 55. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is High) (0.3 (
Rule 56. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Normal) then (output1 is High) (0.3(
Rule 57. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 58. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Sensitive) then (output1 is High) (0.3 (
Rule 59. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Normal) then (output1 is High) (0.3(
Rule 60. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Non-Sensitive) then (output1 is High) (0.3(
Rule 61. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is High) (0.3 (
Rule 62. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Normal) then (output1 is High) (0.3 (
Rule 63. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is High) (0.3 (
Rule 64. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is Medium) (0.3 (
Rule 65. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (
Rule 66. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 67. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Normal) and (Destination_Account_Tries is Sensitive) then (output1 is High) (0.3(
Rule 68. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Normal) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (
Rule 69. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Normal) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 70. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is High) (0.3 (

Rule 71. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Normal) then (output1 is High) (0.3(
Rule 72. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Normal) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is High) (0.3(
Rule 73. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is Medium) (0.3 (
Rule 74. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (

Rule 75. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Non-Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Low) (0.3 (
Rule 76. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Sensitive) then (output1 is Medium) (0.3 (

Rule 77. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (
Rule 78. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Normal) and (Destination_Account_Tries is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 79. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Sensitive) then (output1 is High) (0.3 (

Rule 80. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Normal) then (output1 is Medium) (0.3 (

Rule 81. If (Transaction_Notes is Sensitive) and (Destination_Profile_ID is Non-Sensitive) and (Password_Tries is Sensitive) and (Destination_Account_Tries is Non-Sensitive) then (output1 is High) (0.3)

Number of Rules: 81

Rules for Layer 3 (Environment Segment)

There are 27 rules extracted from MATLAB fuzzy set, layer 3 represent environment segment.

Fuzzy Rules:

Rule 1. If (Daily_Transactions is Sensitive) and (Transaction_Time is Non-Sensitive) and (Time_On_Site is Sensitive) then (output1 is Medium) (0.3 (
Rule 2. If (Daily_Transactions is Sensitive) and (Transaction_Time is Non-Sensitive) and (Time_On_Site is Normal) then (output1 is Medium) (0.3(
Rule 3. If (Daily_Transactions is Sensitive) and (Transaction_Time is Non-Sensitive) and (Time_On_Site is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 4. If (Daily_Transactions is Sensitive) and (Transaction_Time is Normal) and (Time_On_Site is Sensitive) then (output1 is Medium) (0.3 (
Rule 5. If (Daily_Transactions is Sensitive) and (Transaction_Time is Normal) and (Time_On_Site is Normal) then (output1 is High) (0.3 (
Rule 6. If (Daily_Transactions is Sensitive) and (Transaction_Time is Normal) and (Time_On_Site is Non-Sensitive) then (output1 is High) (0.3 (
Rule 7. If (Daily_Transactions is Sensitive) and (Transaction_Time is Sensitive) and (Time_On_Site is Sensitive) then (output1 is High) (0.3 (
Rule 8. If (Daily_Transactions is Sensitive) and (Transaction_Time is Sensitive) and (Time_On_Site is Normal) then (output1 is High) (0.3(
Rule 9. If (Daily_Transactions is Sensitive) and (Transaction_Time is Sensitive) and (Time_On_Site is Non-Sensitive) then (output1 is High) (0.3 (
Rule 10. If (Daily_Transactions is Normal) and (Transaction_Time is Non-Sensitive) and (Time_On_Site is Sensitive) then (output1 is Low) (0.3 (
Rule 11. If (Daily_Transactions is Normal) and (Transaction_Time is Non-Sensitive) and (Time_On_Site is Normal) then (output1 is Medium) (0.3 (
Rule 12. If (Daily_Transactions is Normal) and (Transaction_Time is Non-Sensitive) and (Time_On_Site is Non-Sensitive) then (output1 is Medium) (0.3 (

Rule 13. If (Daily_Transactions is Normal) and (Transaction_Time is Normal) and (Time_On_Site is Sensitive) then (output1 is Low) (0.3 (
Rule 14. If (Daily_Transactions is Normal) and (Transaction_Time is Normal) and (Time_On_Site is Normal) then (output1 is Medium) (0.3 (
Rule 15. If (Daily_Transactions is Normal) and (Transaction_Time is Normal) and (Time_On_Site is Non-Sensitive) then (output1 is High) (0.3 (
Rule 16. If (Daily_Transactions is Normal) and (Transaction_Time is Sensitive) and (Time_On_Site is Sensitive) then (output1 is Medium) (0.3 (
Rule 17. If (Daily_Transactions is Normal) and (Transaction_Time is Sensitive) and (Time_On_Site is Normal) then (output1 is Medium) (0.3(
Rule 18. If (Daily_Transactions is Normal) and (Transaction_Time is Sensitive) and (Time_On_Site is Non-Sensitive) then (output1 is High) (0.3 (
Rule 19. If (Daily_Transactions is Non-Sensitive) and (Transaction_Time is Non-Sensitive) and (Time_On_Site is Sensitive) then (output1 is Low) (0.3 (
Rule 20. If (Daily_Transactions is Non-Sensitive) and (Transaction_Time is Non-Sensitive) and (Time_On_Site is Normal) then (output1 is Low) (0.3 (
Rule 21. If (Daily_Transactions is Non-Sensitive) and (Transaction_Time is Non-Sensitive) and (Time_On_Site is Non-Sensitive) then (output1 is Medium) (0.3(
Rule 22. If (Daily_Transactions is Non-Sensitive) and (Transaction_Time is Normal) and (Time_On_Site is Sensitive) then (output1 is Low) (0.3 (
Rule 23. If (Daily_Transactions is Non-Sensitive) and (Transaction_Time is Normal) and (Time_On_Site is Normal) then (output1 is Medium) (0.3 (
Rule 24. If (Daily_Transactions is Non-Sensitive) and (Transaction_Time is Normal) and (Time_On_Site is Non-Sensitive) then (output1 is Medium) (0.3 (
Rule 25. If (Daily_Transactions is Non-Sensitive) and (Transaction_Time is Sensitive) and (Time_On_Site is Sensitive) then (output1 is Medium) (0.3 (
Rule 26. If (Daily_Transactions is Non-Sensitive) and (Transaction_Time is Sensitive) and (Time_On_Site is Normal) then (output1 is Medium) (0.3 (
Rule 27. If (Daily_Transactions is Non-Sensitive) and (Transaction_Time is Sensitive) and (Time_On_Site is Non-Sensitive) then (output1 is High) (0.3)

Number of Rules: 27

Appendix B

Sample extracted data used in experiments

Sample of data used in our real evaluation experiments. Data is extracted from the internet banking system used in Jordan Ahli Bank. Data reflects a specific period of time.

- **Sample extracted data for Layer 1 (before and after mapping)**

Table B-1: extracted data to be processed for layer 1 (before mapping)

Account_Currency	**TO_Account_Currency**	Transaction ID	**Transaction_Amount**	**Transaction_Currency**	**Same/To Same Curr**
02	01	123456	122.22	USD	No
01	01	61520	100	JOD	Yes
01	01	61653	210	JOD	Yes
01	01	62559	104	JOD	Yes
01	01	62612	300	JOD	Yes
01	02	62706	800	JOD	No
01	01	62782	50	JOD	Yes
01	01	62937	7550	JOD	Yes
01	01	47307	100	JOD	Yes
01	01	63873	1.725	JOD	Yes
01	01	72931	80	JOD	Yes
01	01	73210	303	JOD	Yes
01	02	73355	1455	JOD	No
01	01	73891	30	JOD	Yes
01	01	73925	20000	JOD	Yes
01	01	74112	250	JOD	Yes
01	01	74820	100	JOD	Yes
01	01	75153	245	JOD	Yes
03	01	75812	100	EUR	No
01	01	75973	125	JOD	Yes
01	01	76096	1000	JOD	Yes
01	01	76175	60	JOD	Yes
01	02	76286	500	JOD	No
01	02	76296	450	JOD	No
01	01	76350	10000	JOD	Yes
01	01	77327	110	JOD	Yes

Table B-2: extracted data for layer 1 (after mapping)

Transaction Amount	**Transaction Currency**	**Account Type**	**Account Layer Importance Level Rate**
Non-Sensitive	Non-Sensitive	Non-Sensitive	**Low**
Non-Sensitive	Non-Sensitive	Normal	**Low**
Normal	Normal	Sensitive	**Medium**
Normal	Normal	Normal	**Medium**
Sensitive	Non-Sensitive	Sensitive	**High**
Normal	Non-Sensitive	Sensitive	**Medium**
Sensitive	Non-Sensitive	Non-Sensitive	**Low**
Non-Sensitive	Sensitive	Non-Sensitive	**Low**
Non-Sensitive	Normal	Normal	**Low**
Non-Sensitive	Sensitive	Non-Sensitive	**Low**
Sensitive	Sensitive	Non-Sensitive	**High**
Normal	Sensitive	Normal	**Medium**
Non-Sensitive	Sensitive	Normal	**Low**
Non-Sensitive	Normal	Non-Sensitive	**Low**
Non-Sensitive	Non-Sensitive	Sensitive	**Low**
Non-Sensitive	Non-Sensitive	Non-Sensitive	**Low**
Non-Sensitive	Normal	Sensitive	**Low**
Normal	Sensitive	Non-Sensitive	**Low**
Non-Sensitive	Non-Sensitive	Non-Sensitive	**Low**
Sensitive	Non-Sensitive	Non-Sensitive	**Low**
Sensitive	Sensitive	Non-Sensitive	**High**
Normal	Non-Sensitive	Sensitive	**Low**
Sensitive	Sensitive	Normal	**High**
Sensitive	Non-Sensitive	Normal	**Low**
Sensitive	Non-Sensitive	Non-Sensitive	**Low**
Normal	Non-Sensitive	Non-Sensitive	**Low**
Sensitive	Sensitive	Non-Sensitive	**High**
Non-Sensitive	Non-Sensitive	Non-Sensitive	**Low**
Non-Sensitive	Non-Sensitive	Normal	**Low**
Sensitive	Sensitive	Normal	**High**
Normal	Sensitive	Non-Sensitive	**Low**
Non-Sensitive	Non-Sensitive	Normal	**Low**
Non-Sensitive	Normal	Sensitive	**Low**
Non-Sensitive	Normal	Non-Sensitive	**Low**
Sensitive	Sensitive	Non-Sensitive	**High**
Sensitive	Normal	Normal	**Medium**

- **Sample extracted data for Layer 2 (before and after mapping)**

Table B-3: extracted data for layer 2 (before mapping)

Transaction ID	**Account_Type**	**Transaction_CODE**	**Destination_ProfileID**	**Destination_Account_Tries**	**Incorrect_Password_Tries**
123456	002	00	962002	3	0
61520	001	00	962001	3	0
61653	002	00	962002	1	0
62559	001	00	962001	2	1
62612	001	00	962001	2	1
62706	001	00	962001	2	1
62782	001	00	962001	2	1
62937	002	00	962002	2	0
47307	001	00	962001	2	0
63873	001	00	962001	2	0
72931	003	02	962001	2	0
73210	001	00	962001	2	0
73355	001	00	962001	2	0
73891	001	00	962001	2	0
73925	001	00	962001	2	0
74112	001	00	962001	2	0
74820	001	00	962001	2	0
75153	001	00	962001	2	0
75812	001	00	962001	2	0
75973	001	00	962001	2	0
76096	001	00	962001	2	0
76175	001	00	962001	1	0
76286	002	00	962002	1	0
76296	002	00	962002	1	0
76350	001	00	962001	1	0
77327	001	00	962001	3	0

Table B-4: extracted data for layer 2 (after mapping)

Transaction Notes	**Profile ID**	**Account Tries**	**Incorrect Password Tries**	**Details Layer Importance Level Rate**
Normal	Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Sensitive	Non-Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Non-	Normal	Non-Sensitive	Normal	**Low**

Sensitive				
Non-Sensitive	Normal	Sensitive	Sensitive	**High**
Non-Sensitive	Sensitive	Sensitive	Sensitive	**High**
Normal	Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Normal	Non-Sensitive	Normal	Non-Sensitive	**Low**
Normal	Non-Sensitive	Normal	Non-Sensitive	**Low**
Normal	Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Normal	Non-Sensitive	Non-Sensitive	Non-Sensitive	**Low**
Non-Sensitive	Normal	Sensitive	Sensitive	**High**
Non-Sensitive	Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Sensitive	Non-Sensitive	Sensitive	Sensitive	**High**
Normal	Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Sensitive	Normal	Sensitive	Normal	**High**
Non-Sensitive	Normal	Sensitive	Sensitive	**High**
Non-Sensitive	Normal	Sensitive	Non-Sensitive	**Medium**
Non-Sensitive	Non-Sensitive	Sensitive	Normal	**Medium**
Sensitive	Non-Sensitive	Sensitive	Sensitive	**High**
Non-Sensitive	Normal	Sensitive	Sensitive	**High**
Sensitive	Normal	Sensitive	Normal	**High**
Sensitive	Normal	Sensitive	Sensitive	**High**
Non-Sensitive	Normal	Sensitive	Non-Sensitive	**Medium**
Sensitive	Normal	Sensitive	Sensitive	**High**
Normal	Non-Sensitive	Normal	Sensitive	**Medium**
Non-Sensitive	Non-Sensitive	Sensitive	Normal	**Medium**
Normal	Non-Sensitive	Normal	Sensitive	**Medium**
Non-Sensitive	Normal	Normal	Sensitive	**Medium**
Sensitive	Sensitive	Non-Sensitive	Sensitive	**High**
Non-Sensitive	Normal	Sensitive	Non-Sensitive	**Medium**
Normal	Non-Sensitive	Normal	Non-Sensitive	**Low**

Non-Sensitive	Normal	Sensitive	Sensitive	**High**
Normal	Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Sensitive	Sensitive	Normal	Normal	**High**
Non-Sensitive	Normal	Sensitive	Sensitive	**High**
Non-Sensitive	Normal	Sensitive	Sensitive	**High**

Table B-5: extracted data for layer 3 (before mapping)

Transaction ID	**Sequence_NO**	**Posting_Time**	**Logged_IP**
123456	01	7:07:50	128.100.22.121
61520	01	8:46:37	77.241.64.34
61653	01	9:54:20	77.245.0.12
62559	01	9:57:21	79.134.128.64
62612	01	10:05:44	79.173.192.11
62706	01	10:11:11	80.64.208.112
62782	01	10:24:06	80.90.160.93.1
62937	01	10:33:02	80.249.208.108.6
47307	01	11:47:16	81.28.112.124.1
63873	01	7:58:08	82.212.64.139.6
72931	01	9:00:46	84.18.32.155.1
73210	01	9:15:45	84.18.64.170.6
73355	01	10:07:49	86.108.0.186.1
73891	01	10:09:59	91.186.224.201.6
73925	01	10:32:11	92.62.112.217.1
74112	01	12:14:24	92.241.32.232.6
74820	01	1:05:35	94.142.32.248.1
75153	01	2:46:01	94.249.0.263.6
75812	01	3:25:49	95.140.160.75
75973	01	3:40:17	95.141.208.43
76096	01	3:54:27	95.172.192.32
76175	01	4:11:19	188.123.128.112
76286	01	4:12:27	188.123.160.90.5
76296	01	4:18:45	188.244.96.100.5
76350	01	12:19:27	188.247.64.110.5
77327	01	7:25:19	193.188.64.120.5

Table B-6: extracted data for layer 3 (after mapping)

Time on Service	Daily Transactions	Transaction Time	Environment Layer Importance Level Rate
Sensitive	Normal	Sensitive	**High**
Non-Sensitive	Normal	Sensitive	**High**
Non-Sensitive	Sensitive	Sensitive	**High**
Normal	Non-Sensitive	Normal	**Medium**
Sensitive	Non-Sensitive	Sensitive	**High**
Non-Sensitive	Normal	Sensitive	**High**
Non-Sensitive	Sensitive	Sensitive	**Medium**
Normal	Sensitive	Non-Sensitive	**Medium**
Sensitive	Non-Sensitive	Sensitive	**High**
Normal	Non-Sensitive	Normal	**Low**
Non-Sensitive	Non-Sensitive	Sensitive	**Medium**
Normal	Sensitive	Non-Sensitive	**Medium**
Normal	Non-Sensitive	Normal	**High**
Non-Sensitive	Non-Sensitive	Sensitive	**Medium**
Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Sensitive	Normal	Normal	**Medium**
Normal	Non-Sensitive	Normal	**Medium**
Non-Sensitive	Normal	Sensitive	**High**
Sensitive	Normal	Sensitive	**High**
Sensitive	Normal	Sensitive	**High**
Non-Sensitive	Sensitive	Sensitive	**High**
Non-Sensitive	Sensitive	Sensitive	**Medium**
Non-Sensitive	Sensitive	Sensitive	**High**
Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Non-Sensitive	Normal	Non-Sensitive	**Low**
Non-Sensitive	Normal	Sensitive	**Medium**
Non-Sensitive	Non-Sensitive	Sensitive	**Low**
Non-Sensitive	Normal	Sensitive	**High**
Sensitive	Normal	Sensitive	**High**
Normal	Non-Sensitive	Non-Sensitive	**Low**
Non-Sensitive	Non-Sensitive	Sensitive	**Medium**
Non-Sensitive	Sensitive	Non-Sensitive	**Low**
Normal	Non-Sensitive	Normal	**Medium**

Appendix C

Sample XML messages and sample DTDs

- **Sample Source XML Messages**

```
<?xml version='1.0'?>
<Transfers>
<Transaction xmlns='http://example.org/paymentv2' ImportanceLevel="000">
    <Transaction_Notes>002</Transaction_Notes>
    <Profile_ID>92</Profile_ID>
    <AccountTries>02</AccountTries>
    <PasswordTries>01</PasswordTries>
</Transaction>

<Transaction xmlns='http://example.org/paymentv2' ImportanceLevel="000">
    <Posting_Date>2011/01/07</Posting_Date>
    <Service_ID>WWW60</Service_ID>
    <Customer_Language>E</Customer_Language>
</Transaction>

<Transaction xmlns='http://example.org/paymentv2' ImportanceLevel="002">
    <TransactionAmount>755</TransactionAmount>
    <Transaction_Currency Code='JOD'>001</Transaction_Currency>
    <Account_Type Code='001'>Individual</Account_Type>
</Transaction>

<Transaction xmlns='http://example.org/paymentv2' ImportanceLevel="001">
    <IPAddress>128.200.3.212</IPAddress>
    <From_Account>390230101401043000</From_Account>
    <To_Account>120130101155414000</To_Account>
</Transaction>

</Transfers>
```

- **Sample Encrypted XML Messages**

```
<?xml version='1.0'?>
<Transfers>
```

```
<Transaction xmlns='http://example.org/paymentv2' ImportanceLevel="Low">
        <Transaction_Notes>002</Transaction_Notes>
        <Profile_ID>92</Profile_ID>
        <AccountTries>02</AccountTries>
        <PasswordTries>01</PasswordTries>
</Transaction>

<Transaction xmlns='http://example.org/paymentv2' ImportanceLevel="Low">
        <Posting_Date>2011/01/07</Posting_Date>
        <Service_ID>WWW60</Service_ID>
        <Customer_Language>E</Customer_Language>
</Transaction>

<Transaction xmlns='http://example.org/paymentv2' ImportanceLevel="Medium">
        <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
         xmlns='http://www.w3.org/2001/04/xmlenc#'>
                <CipherData>
                        <CipherValue>i66xTe5hmF/siLFCXDPXTucE9wJZFd
                        pbSV3yYwN7pBZKIOTdRdD6LtOgunRVgjxKirSWpLx0yZ3l
                        0KoZS9B1gKOalZUjE8Sp8Al8qKgrxbfx3CR7fIdEKPdO47t6hr
                        swwL7lewxZnrJo5Whd2kw/XRUb
                        uXp268jX5dL0dlUDOEqdtgfPUXxROUetbLP1AmtO8riJWVh/
                        Qyd3pvVZtNOn9mo0CbclDn0UsntXHFEfst8=
                        </CipherValue>
                </CipherData>
</EncryptedData>
</Transaction>

<Transaction xmlns='http://example.org/paymentv2' ImportanceLevel="High">
<EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
xmlns='http://www.w3.org/2001/04/xmlenc#'>
        <CipherData>
                <CipherValue>6N28UemsjUz4vegVtDE1wNNgNkvTvC6Pxi9k2vcHcG
                IhSeH+kUIdd2IWvb/62gepoBCDhFGI+XQ9
                DWGvHlebDHA7DgNjmm+D37lU1DuzsB094b8cUPZH9gCjX0VDRn
                tfDTFyeiUtLUKIdH1vUi/m+ok/
                pxnBFo35XheoJQFcLv21Kxz7Tzffh9ZCaqYU8HBE
                </CipherValue>
        </CipherData>
</EncryptedData>
</Transaction>

</Transfers>
```