Intelligent XML Encryption using Effective Fuzzy Classification

Abstract: **In this paper we present a novel approach for securing financial XML transactions using an effective and intelligent fuzzy classification technique. 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 "ImportanceLevel". Assigned value indicates the data sensitivity for each XML tag. The framework 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 one of the leading banks in Jordan to demonstrate its flexibility, feasibility, and efficiency. Our experimental results of the new model verified tangible enhancements in encryption efficiency, processing-time reduction, and resulting XML message sizes.

***Index Terms* –XML Encryption, Fuzzy XML, Fuzzy Classification, XML Security, Banking Security.**

I. INTRODUCTION

The eXtensible Markup Language (XML) [1] 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 [2]. 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. There are several approaches proposed by researchers to secure XML messages.

Many models have been proposed to protect exchanged messages both on the network level [10, 11] and on the XML level. Among the proposed models, W3C played a major role, providing standardized forms to represent XML data in a secure and trusted method. W3C introduced XML Encryption [3], XML Signature [4], and XML Key Management [5].

The XML Encryption standard defines how to encrypt the XML message. This can involve fully encrypting the entire message, partially encrypting it by selecting parts of each message, or even encrypting external elements attached to the message itself. Although this model is able to secure XML messages, some issues arose concerning performance and inefficient memory usage [12, 13], leaving room for more improvements and enhancements.

However, financial institutions (i.e. banks) perform large volume of transactions on daily basis which require XML encryption on large scale. Encrypting large volume of messages in full will result performance and resource issues. Therefore, an approach is needed to encrypt specified portions of an XML document, syntax for representing encrypted parts, and processing rules for decrypting them. W3C XML encryption has the feature to encrypt parts of an XML document called element-wise encryption which is the process of encrypting parts of the XML document. To avoid any performance or resources issues, a mechanism should be considered to choose which parts of the XML document to be encrypted on the fly, whereby those parts are selected upon intelligent criteria detecting sensitive information within the XML document.

Fuzzy Logic (FL) [18] approach can be used here to distinguish sensitive parts within each XML document. FL provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. FL's approach to control problems mimics how a person would make faster decision. FL incorporates a simple, rule-based 'IF X AND Y THEN Z' approach to a solving control problem rather than attempting to model a system mathematically. The FL model is empirically-based, relying on an operator's experience rather than their technical understanding of the system.

Fuzzy logic approach is quantified based on a combination of historical data and expert input. Fuzzy logic has been used for decades in the computer sciences to embed expert input into computer models for a broad range of applications. The advantage of the fuzzy approach is that it enables processing of vaguely defined variables, and variables whose relationships cannot be defined by mathematical relationships. Fuzzy logic can incorporate expert human judgment to define those variables and their relationships. The model can be closer to reality and be more site specific than some of the other methods [19].

II. LITERATURE REVIEW

Flexibility, expressiveness, and usability of XML have formed a motive for researchers to shed more light on XML security. Researchers have focused their interests on securing XML data due to the increased usage of XML in many business and educational cases. Efficient models have been proposed [3, 4, 5, 9, 10, 11] to add a secure layer over exchanged XML data. The models' main purpose is to ensure data confidentiality and authenticity. Many XML threats [12] have been considered, such as Oversized Payload, Schema Change, XML Routing, and Recursive Payload. Such threats have forced researchers to pay more attention to securing exchanged XML messages.

W3C XML Encryption Working Group [3] is developing a method for XML encryption and decryption. The group used XML syntax to represent the secured elements in XML. Their approach is able to encrypt the whole message, full nodes, and sub-trees; however, it is not able to encrypt an element while keeping the descendants of the same node unchanged, and also it cannot handle attribute encryption. Therefore, a solution has been proposed [13] to handle this limitation. Ed Simon proposed changing the attribute so that it is encrypted with the *EncryptedDataManifest* attribute and including any other details inside the element. Another solution proposed was to use XSLT for attribute transformation into elements to perform the encryption process. However, this suggested solution did not face success, as the decrypted parts need to be transformed back to the original attributes for message validation against the corresponding XML schema.

A system has been proposed by [15] for pool encryption, which has the capability of removing sensitive information from the output file. Their basic idea is to parse the XML message which needs encryption into a DOM tree, where each node in the tree is labeled and all information related to its position is attached to the corresponding node. Then each node is encrypted individually with a "node specific" encryption key. These nodes are removed from their original position in the XML message into a pool which contains all other encrypted nodes. The pool can be saved into the original message or in a different message. The sender determines the decryption capabilities of different users by distributing the collection of node keys to the receiver. This collection of node keys is encrypted with the recipients' key before final submission. Although this model solves the issue of removing confidential material from the main message and hides the size of the encrypted content, it has the following disadvantages:
The original position for each individual node needs to be attached, Due to the addition of "the position information", a decent increase in message size is noticed, Due to the pool of node keys, a decent increase in message size is noticed, and High resource usage and bandwidth allocation, more storage more processing power is needed, and A unique node key has to be generated for each node.

[20] Introduced an XML access control (XAC) that is a server-side access control and a trusted access control processor allowing security policies and procedures to be established based on the policies, XAC present a way to control access of users to specific portions of the full XML document that is stored on a server. XAC encrypts an XML element with the ability to exclude its descendants. This specific feature gives the advantage of XAC over XEnc because XEnc requires the encryption of a full sub-tree.

[21] Presented an approach to incorporate fuzziness in XML. Their approach tried to identify the potential entities in XML that can have fuzzy values. They analyze the structure of an XML document to identify the portions that can be handled using fuzziness; then they specify the appropriate mechanism to incorporate fuzziness. Their approach focused on XML being structured (logical and physical) and well-formed language.

[22] Introduced a fuzzy XML data model to manage fuzzy data in XML, based on possibility distribution theory, by first identified multiple granularity of data fuzziness in UML and XML. The fuzzy UML data model and fuzzy XML data model that address all types of fuzziness are developed. Further, they developed the formal conversions from the fuzzy UML model to the fuzzy XML model, and the formal mapping from the fuzzy XML model to the fuzzy relational databases.

[23] 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 stylesheets. With an example, they represent that given fuzzy system in XML and transform it to comprehensible formats for Matlab and FuzzyJess applications.

[24] Proposed an approach along with an automated tool called (FXML2FOnto) for constructing fuzzy ontologies from fuzzy XML models, they also investigated how the constructive fuzzy ontologies may be useful for improving some of the fuzzy XML applications (i.e. reasoning on fuzzy XML models).

III. System Model and Design

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 [8] and encrypted using an element-wise encryption on selected parts within each XML message. The fuzzification phase is performed before the XML messages are submitted to the next phase which is responsible for securing message content. The process of fuzzy classification is mainly responsible for defining an attribute value and assigning it to an existing XML tag named "ImportanceLevel". The assigned value will be used to define the security level needed in the next phase. Next phase 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 fuzzification phase 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 [3].

Figure 1.0 illustrates the system model and basic components used to form our framework. As seen in the figure, the main two components are displayed as two separate units each act as an independent unit performing set of operations that used as input to the other phase.

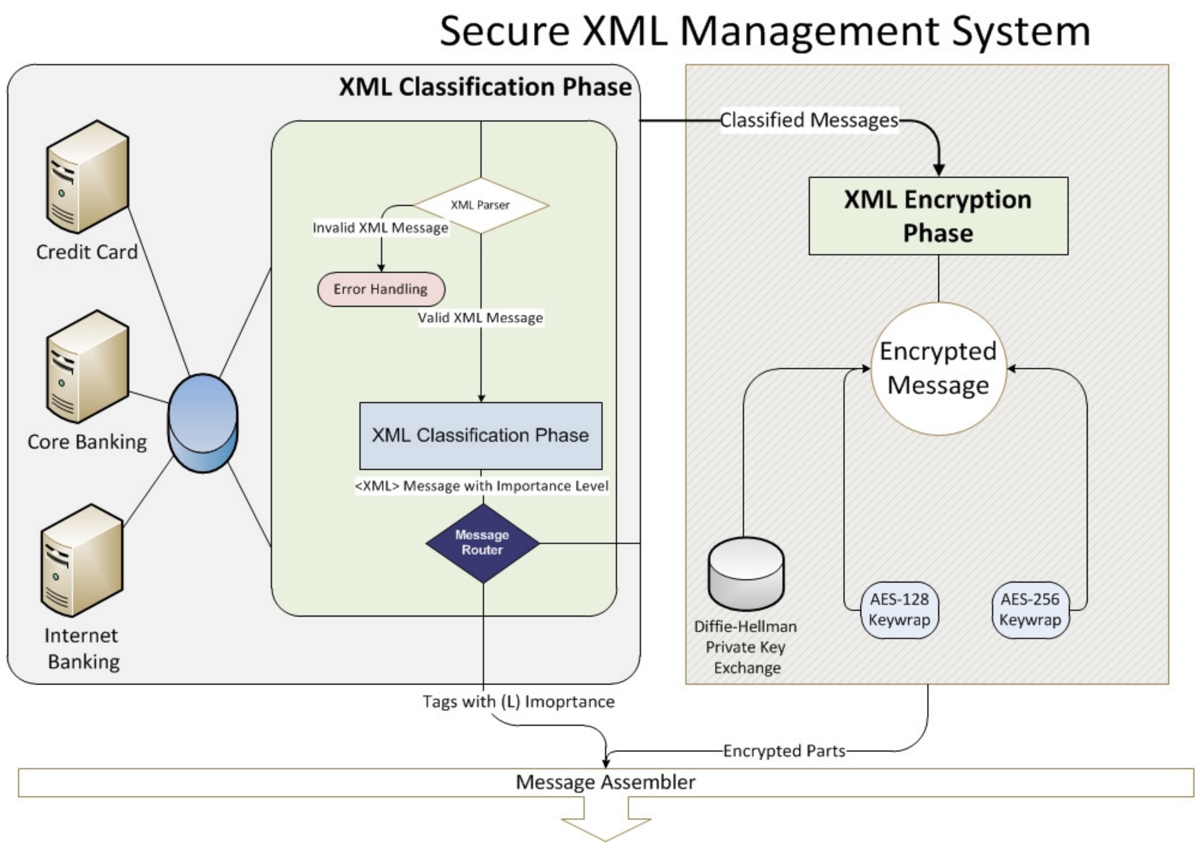


Figure 1: Main System Design

The system core has been built based on two major phases. Phase one involves performing a set of fuzzy classification techniques on XML messages. The fuzzy classification process is designed mainly for deciding the similarity of different standards within the same message. Basically the main target is to describe how semantic concepts are evaluated and explained by the provided XML content. Upon fuzzy classification, a new value is generated and assigned to an existing XML tag. We assigned the name "ImportanceLevel" to the mentioned tag so we can use it as an identifier for the next phase. Phase two involves applying element-wise encryption to different parts within each XML message. Encryption could be for the whole message, some elements, or some attributes of an element of an XML message. The ImportanceLevel value assigned in phase one is used to decide which type of encryption is performed, and also decides which parts of the XML message are to be encrypted. We base our encryption on W3C's recommendation [3]. The following stages form system life cycle in details:

1. Fuzzy Classification Phase

In our fuzzy classification phase, 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. The architecture of the fuzzy logic inference-based classification model is shown in Figure 2.

This phase is responsible for assigning a new value which is the importance level for each XML tag. The main idea is distinguish which parts of the message is to be encrypted using AES-128 bit key encryption and which are to be encrypted using AES-256 bit key. Usage of the key depends on the importance level value (high, medium or low), whereby we deploy the 128-bit key on tags with "Medium" Importance Level and the 256-bit key on tags with "High" Importance Level value. Tags with a "Low" importance level value are forwarded directly to the message assembler where no encryption is performed. The phase uses fuzzification techniques of a set of input variables based on ten 10 characteristics extracted from the XML message, all depending on the previous knowledge experience and expertise backgrounds. The 10 characteristics are defined in details 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 total transaction performed. Source, destination, and amount all combined to act as an alert which is already pre-defined based on 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: 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 are with large volume which can benefit the bank for each transaction. We used this factor due to its role 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 term of transaction 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 ID's have a high potential to be monitored for transactions, monitoring is based on the transaction amount after classifying each profile id whereby a range of ID's are listed in the high importance zone, all after deploying 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 regular basis to confirm trusted accounts and suspicious ones. Evaluation will result a set of important ranges of trusted accounts to be used in transaction evaluation and setting 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 importance level for each transaction, high rate of incorrect tries gives 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 considered a technical factors to measure 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. Number of daily transactions put a weight on overall importance level for the transaction itself, whereby number of transactions to be performed is set based on 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 an importance level 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.

2. Fuzzy Methodology

Our fuzzy classification phase is based on Mamdani [8] fuzzy inference, performing the basic four steps shown in Figure 2.

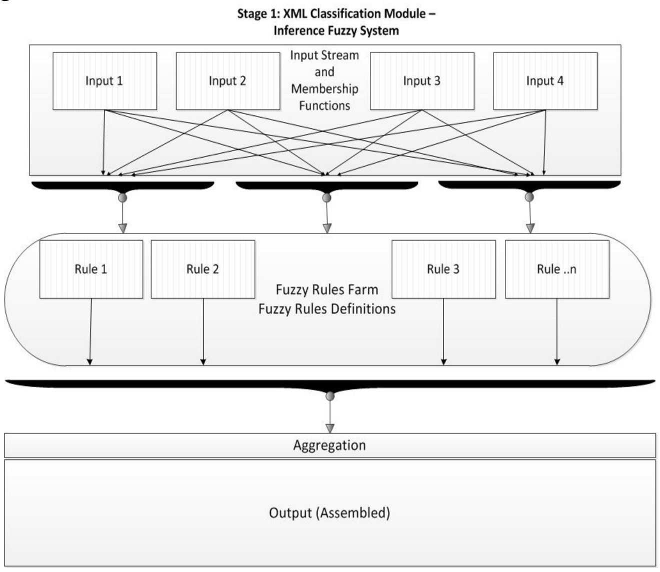


Figure 2 Mamdani fuzzy inference system

Step 1 (Fuzzification): Take the crisp input X and Input Y and determine the degree to which these inputs belong to, and where they fit into, the fuzzy set. Figure 3 illustrates an example of a linguistic variable used representing one factor, which is the transaction currency. The x-axis represents the range of transaction amount. The y-axis represents the degree of each value in the linguistic descriptor.

Transaction Currency (Non-Sensitive, Normal, Sensitive)
Variable used: Transaction Amount
Ranges:

Non-Sensitive:	[0, 0, 6, 8]
Normal:	[6, 9, 12]
Sensitive:	[10, 12, 18, 18]

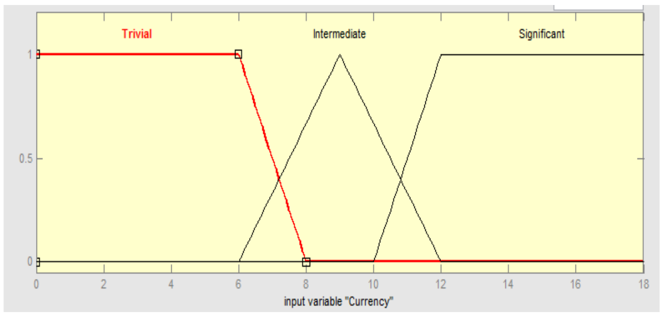


Figure 3 Input variables in the fuzzification step

Step 2 (Rule Evaluation): Take the fuzzy inputs and apply them to the qualified fuzzy rules. The fuzzy operators (AND / OR) are used in case of any uncertainty to get a single value. The outcome value is called "Truth Value" which will be applied to the membership function for rule evaluation.
Step 3 (Aggregation of the Rule Outputs): Process of unification of the outputs of all the rules. Combining scaled rules into a single fuzzy set for each variable.
Step 4 (Transforming the fuzzy output into a crisp output): Figure 4 illustrates an example of an expected crisp output [Low, Medium, and High]

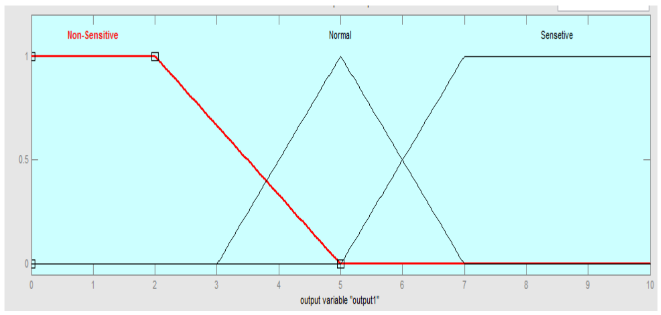


Figure 4 Sample output of classification rate "importance level"

The output should have a clear crisp value where it will be assigned to each tag classified.

Low: Means the importance level is low and more attention should not be paid to the value. The root element and child tags should be forwarded directly to the message assembler, skipping the encryption phase.

Medium: The tag is important to some extent, and the tag attribute is assigned the value of medium so an element-wise encryption will be applied using the AES algorithm with a 128-bit key on selected parts.

High: To be handled with high importance and encrypted in the next phase using the AES algorithm with a 256-bit key.

3. Detection Module

To perform the fuzzy inference system we have categorized the XML tags within each message into 10 characteristics distributed into three layers, each with its own weight and criteria. The layers are Account Layer, Details Layer, and Environment Layer. Figure 5 represents the layers distribution.

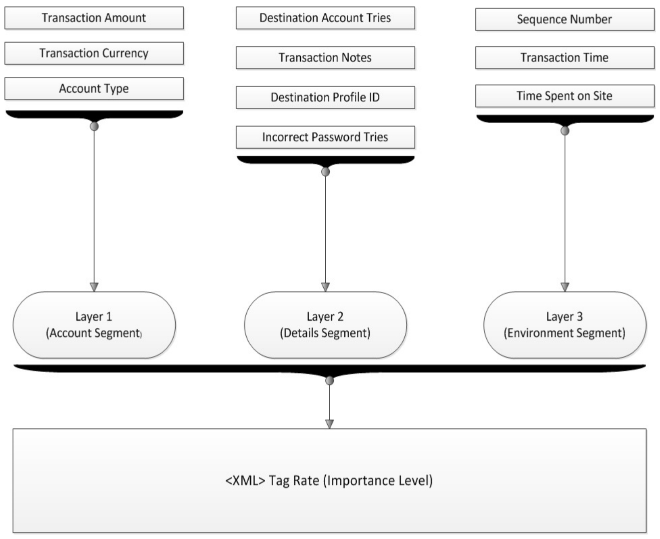


Figure 5 Layers Distribution

By giving a weight to each layer, the calculation of overall weight is based on the following criteria:
Importance Level: **Sum** (Layer Weight * Layer Member)
Rule Base**:** Each layer has a set of rules defined based on input variables within each layer. The rule is based on the "IF-THEN" rule. The rule base should contain a number of entries depending on how many layer members exist. For example, layer 1 has three members and we have three outputs expected, so the entries should be calculated as (3^3) = 27 entries presenting the rules for that layer.
The final evaluation is dependent on finding the centre of gravity as shown in the following equation:

$$COG = \frac{\int \mu_i(x)\ x\ dx}{\int \mu_i(x)\ dx}$$

μi(x): Aggregated membership function.
x: Output variable.
After deploying the fuzzy classification methodology on the three layers, we then have a list of classified tags with an importance level attribute defined and assigned.

4. Encryption Module

The encryption phase has two possibilities: the first one is to perform an element-wise encryption using the AES algorithm with a key size of 256-bit, while the second is to perform an element-wise encryption using the AES algorithm with a key size of 128-bit. Key size is determined by the Importance Level value assigned in the fuzzy classification phase. Figure 6 illustrates the process of encryption. Tags with "Low" ImportanceLevel will be forwarded directly to the message composition stage without any type of encryption being performed.

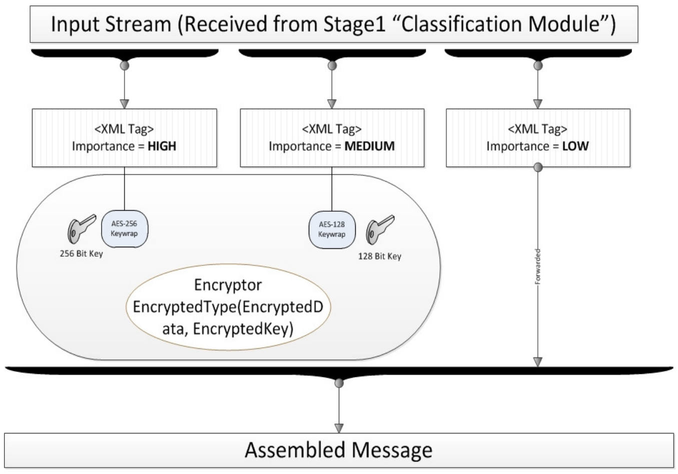
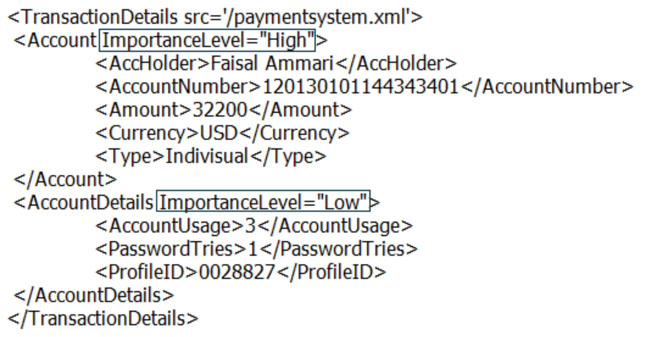

Input Stream (Received from Stage1 "Classification Module")
<XML Tag> Importance = HIGH
<XML Tag> Importance = MEDIUM
<XML Tag> Importance = LOW
AES-256 Keywrap
256 Bit Key
AES-128 Keywrap
128 Bit Key
Encryptor EncryptedType(EncryptedData, EncryptedKey)
Forwarded
Assembled Message

Figure 6 Encryption module layout

Tags related to the parent tag are also encrypted using the same level of encryption. Child tags behaviour is taken from the parent "ImportanceLevel" value.

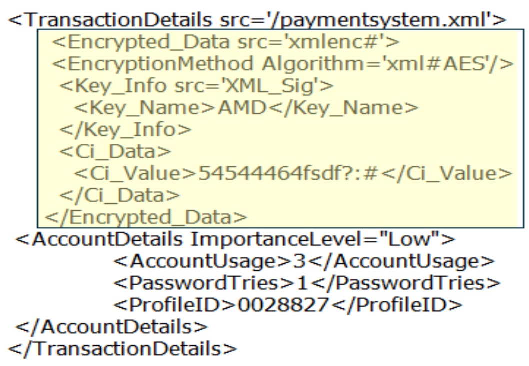
Figure 7 illustrates the XML message after the fuzzy classification phase where the "ImportanceLevel" attribute is assigned a value.



```
<TransactionDetails src='/paymentsystem.xml'>
 <Account ImportanceLevel="High">
            <AccHolder>Faisal Ammari</AccHolder>
            <AccountNumber>120130101144343401</AccountNumber>
            <Amount>32200</Amount>
            <Currency>USD</Currency>
            <Type>Indisivual</Type>
 </Account>
 <AccountDetails ImportanceLevel="Low">
            <AccountUsage>3</AccountUsage>
            <PasswordTries>1</PasswordTries>
            <ProfileID>0028827</ProfileID>
 </AccountDetails>
</TransactionDetails>
```

Figure 7 Sample XML message after fuzzy classification

Figure 8 illustrates the same XML message after encryption depending on the fuzzy classification performed earlier.



```
<TransactionDetails src='/paymentsystem.xml'>
  <Encrypted_Data src='xmlenc#'>
  <EncryptionMethod Algorithm='xml#AES'/>
   <Key_Info src='XML_Sig'>
     <Key_Name>AMD</Key_Name>
   </Key_Info>
   <Ci_Data>
     <Ci_Value>54544464fsdf?:#</Ci_Value>
   </Ci_Data>
  </Encrypted_Data>
 <AccountDetails ImportanceLevel="Low">
           <AccountUsage>3</AccountUsage>
           <PasswordTries>1</PasswordTries>
           <ProfileID>0028827</ProfileID>
 </AccountDetails>
</TransactionDetails>
```

Figure 8 Sample XML message after encryption phase

Tags related to the parent tag are also encrypted using the same level of encryption. Child tags behavior is taken from the parent "ImportanceLevel" value. In Figure 7 (Account Holder, Account Number, Amount, Currency, and Type), tags are encrypted using AES encryption with a key size of 256 bit as per their parent "Account" layer. Basically we inherit the encryption behavior from parent to child as per our categorization process, and the categorization process in our model is built based on relevance and parent tag evaluation.

Keys used during the encryption process should be transferred to the decryptor in the destination using a secure and private method. We use Diffie-Hellman [16] key exchange for the handover of keys between source and destination. Figure 9 illustrates how to exchange keys between source and destination.

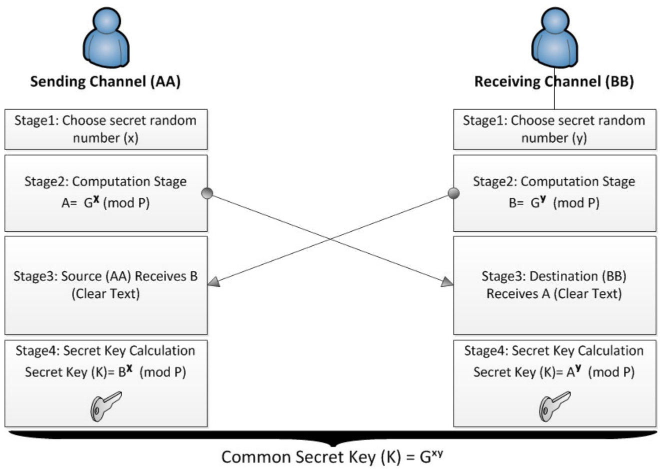

Sending Channel (AA)
Receiving Channel (BB)
Stage1: Choose secret random number (x)
Stage1: Choose secret random number (y)
Stage2: Computation Stage A= G^X (mod P)
Stage2: Computation Stage B= G^Y (mod P)
Stage3: Source (AA) Receives B (Clear Text)
Stage3: Destination (BB) Receives A (Clear Text)
Stage4: Secret Key Calculation Secret Key (K)= B^X (mod P)
Stage4: Secret Key Calculation Secret Key (K)= A^Y (mod P)
Common Secret Key (K) = G^XY

Figure 9 Key exchange using D-H method

IV. Experiment and Results

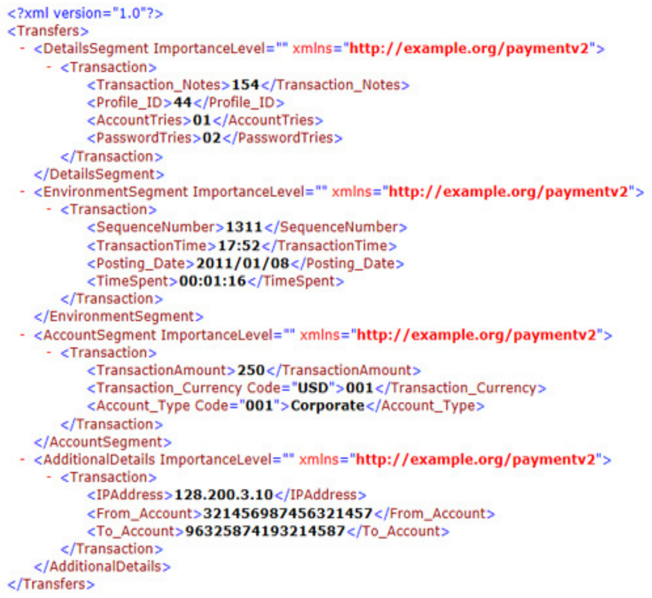
We have performed our evaluation using two sets of XML messages; each set represent a period in which the messages were extracted. Each set has number of XML messages to test. Collected XML messages present online banking service transactions fetched from Jordan Ahli Bank, one of the leading banks in 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 (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 1 illustrates the two sets of XML messages in details.

TABLE 1: Experiment Set details

Set	Messages	Nodes	Size	Period	Encryption
1	1,000	4,000	947 KB	3 Months Jan 12-Mar12	Full
2	1,500	6,000	1380 KB	4 Months Apr12-Aug12	Partial

Figure 10 illustrate an actual XML message fetched form one of the XML messages in set 1.



```
<?xml version="1.0"?>
<Transfers>
 - <DetailsSegment ImportanceLevel="" xmlns="http://example.org/paymentv2">
   - <Transaction>
       <Transaction_Notes>154</Transaction_Notes>
       <Profile_ID>44</Profile_ID>
       <AccountTries>01</AccountTries>
       <PasswordTries>02</PasswordTries>
     </Transaction>
   </DetailsSegment>
 - <EnvironmentSegment ImportanceLevel="" xmlns="http://example.org/paymentv2">
   - <Transaction>
       <SequenceNumber>1311</SequenceNumber>
       <TransactionTime>17:52</TransactionTime>
       <Posting_Date>2011/01/08</Posting_Date>
       <TimeSpent>00:01:16</TimeSpent>
     </Transaction>
   </EnvironmentSegment>
 - <AccountSegment ImportanceLevel="" xmlns="http://example.org/paymentv2">
   - <Transaction>
       <TransactionAmount>250</TransactionAmount>
       <Transaction_Currency Code="USD">001</Transaction_Currency>
       <Account_Type Code="001">Corporate</Account_Type>
     </Transaction>
   </AccountSegment>
 - <AdditionalDetails ImportanceLevel="" xmlns="http://example.org/paymentv2">
   - <Transaction>
       <IPAddress>128.200.3.10</IPAddress>
       <From_Account>321456987456321457</From_Account>
       <To_Account>96325874193214587</To_Account>
     </Transaction>
   </AdditionalDetails>
</Transfers>
```

Figure 10 Actual XML message from Set 1

Table 2, Table 3, and Table 4 illustrate a sample of the data provided in set 1, segregated into three layers.

TABLE 2: SAMPLE OF DATA RECEIVED CLASSIFIED FOR LAYER 1

Transaction Amount	Transaction Currency	Account Type	Account Segment
Non-Sensitive	Non-Sensitive	Non-Sensitive	**Low**
Normal	Normal	Sensitive	**Medium**
Sensitive	Non-Sensitive	Sensitive	**High**
Normal	Non-Sensitive	Sensitive	**Medium**
Sensitive	Non-Sensitive	Non-Sensitive	**Low**

TABLE 3: SAMPLE OF DATA RECEIVED CLASSIFIED FOR LAYER 2

Transaction Notes CODE	Destination ProfileID	Destination Account Tries	Incorrect Password Tries	Details Segment
Normal	Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Sensitive	Non-Sensitive	Non-Sensitive	Non-Sensitive	**Medium**
Non-Sensitive	Normal	Non-Sensitive	Normal	**Low**
Non-Sensitive	Sensitive	Sensitive	Sensitive	**High**
Normal	Sensitive	Non-Sensitive	Non-Sensitive	**Medium**

TABLE 4: SAMPLE OF DATA RECEIVED CLASSIFIED FOR LAYER 3

Time On Site	Daily Transactions	Transaction Time	Transaction Level
Sensitive	Normal	Sensitive	**High**
Non-Sensitive	Sensitive	Sensitive	**High**
Normal	Non-Sensitive	Normal	**Medium**
Sensitive	Non-Sensitive	Sensitive	**High**
Non-Sensitive	Normal	Sensitive	**High**

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 5 illustrates the evaluation details for stage 1.

TABLE 5: STAGE 1 SET DETAILS

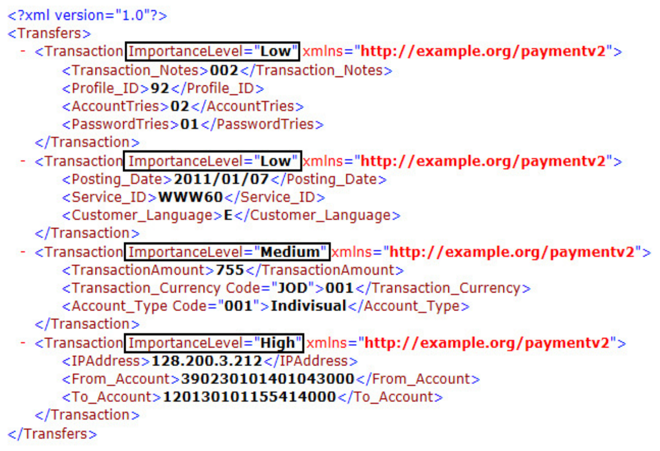
Stage	**XML Messages**	**Model**	**Experiment 1 Used Key**	**Experiment 2 Used Key**
1	**1,000 Messages** 4,000 Nodes	W3C **Full Encryption**	**128 bit**	**256 bit**
		SXMS **Element-Wise**	**128** bit or **256** bit or NO Encryption	**128** bit or **256** bit or NO Encryption

Stage 1: Evaluation for this stage has been conducted by performing two experiments; first experiment deployed by performing full encryption using W3C XML encryption standard with a 128-bit key size, deployed on the first set of 1,000 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. Our model main goal is to optimize and increase encryption-processing time; therefore we have listed the number of occurrences for "High" and "Medium"l importance level which require an encryption process to secure existing content. Table 6 represents the number of occurrences for transactions marked with "High" and "Medium" across the three layers.

TABLE 6: APPEARANCES FOR EACH CLASSIFICATION LAYER

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 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 XML messages 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.
Figure 11 present an actual XML message after fuzzy classification phase where we notice the importance level value assigned per root node in each XML message.



```
<?xml version="1.0"?>
<Transfers>
 - <Transaction ImportanceLevel="Low" xmlns="http://example.org/paymentv2">
      <Transaction_Notes>002</Transaction_Notes>
      <Profile_ID>92</Profile_ID>
      <AccountTries>02</AccountTries>
      <PasswordTries>01</PasswordTries>
   </Transaction>
 - <Transaction ImportanceLevel="Low" xmlns="http://example.org/paymentv2">
      <Posting_Date>2011/01/07</Posting_Date>
      <Service_ID>WWW60</Service_ID>
      <Customer_Language>E</Customer_Language>
   </Transaction>
 - <Transaction ImportanceLevel="Medium" xmlns="http://example.org/paymentv2">
      <TransactionAmount>755</TransactionAmount>
      <Transaction_Currency Code="JOD">001</Transaction_Currency>
      <Account_Type Code="001">Indisivual</Account_Type>
   </Transaction>
 - <Transaction ImportanceLevel="High" xmlns="http://example.org/paymentv2">
      <IPAddress>128.200.3.212</IPAddress>
      <From_Account>390230101401043000</From_Account>
      <To_Account>120130101155414000</To_Account>
   </Transaction>
</Transfers>
```

Figure 11 Classified XML message taken from first implementation

Table 7 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 7: PERFORMANCE EVALUATION FOR STAGE 1 – EXPERIMENT 1

Stage 1 – Experiment 1 (Full Encryption)	**Processing Time**		**File Size**	
XML Message Set	**SXMS Model**	**W3C 128 bit**	**XML Messages**	**SXMS Model**
1 XML File	0.0018 MS	0.0023 MS	1 XML File	0.0018 MS
300 XML	0.562 MS	0.702 MS	300 XML	0.562 MS
600 XML	0.873 MS	1.264 Sec	600 XML	0.873 MS
900 XML	1.271 Sec	1.825 Sec	900 XML	1.271 Sec
1,000 XML (Set 1)	**1.625 Sec**	**2.456 Sec**	**1,000 XML (Set 1)**	**1.625 Sec**

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 measurable 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 7, 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 11 illustrates the comparison between the two models and performance improvement using SXMS.

Table 7 also illustrates files size reduction encrypting XML messages using SXMS model, table shows a measurable 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 measurable amount of space and bandwidth on large scale. Figure 11 illustrates the processing time needed to encrypt the sample messages in the first experiment compared to our model.

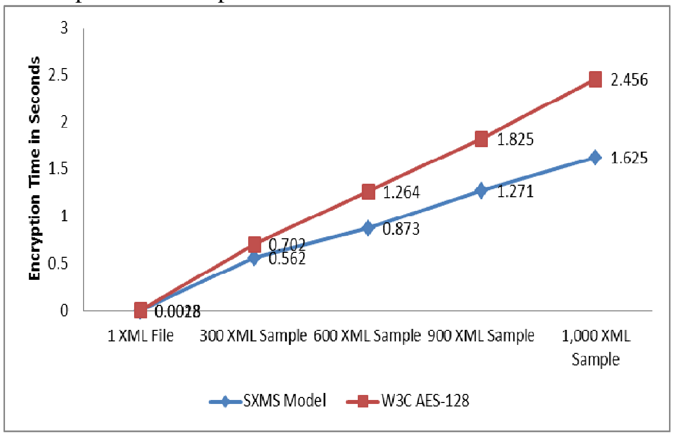


Figure 11 Comparison chart between SXMS and W3C model using 128-bit

As seen in Figure 11, the x-axis present the number of XML messages being processed, while y-axis present the processing time encrypting XML messages in seconds. Figure 12 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.

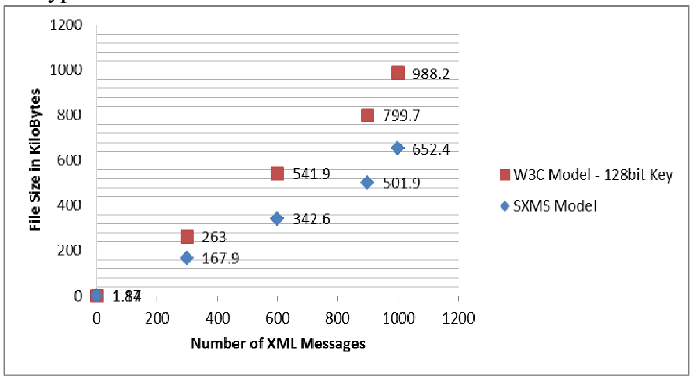


Figure 12 File size comparisons between SXMS and W3C model using 128bit

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 4 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 256 bit encrypting each message in full.

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 measurable improvement in processing time compared to W3C XML encryption model which uses a 256-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”.

TABLE 8: PERFORMANCE EVALUATION FOR STAGE 1 – EXPERIMENT 2

Stage 1 Experiment 2 (Full Encryption)	**Processing Time**		**File Size**	
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	0.562 MS	0.811 MS	167.9 KB	283.4 KB
600 XML	0.873 MS	1.591 Sec	342.6 KB	601 KB
900 XML	1.271 Sec	2.137 Sec	501.9 KB	864.8 KB
1,000 XML	**1.625 Sec**	**2.8 Sec**	**652.4 KB**	**1112 KB**

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.

As seen in Table 8, 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 8 also illustrates files size reduction encrypting XML messages using SXMS model, table shows a measurable 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 measurable amount of space and bandwidth on large scale. Figure 13 illustrates the performance comparison between SXMS model and W3C encryption standard using key size of 256-bit. Figure 14 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.

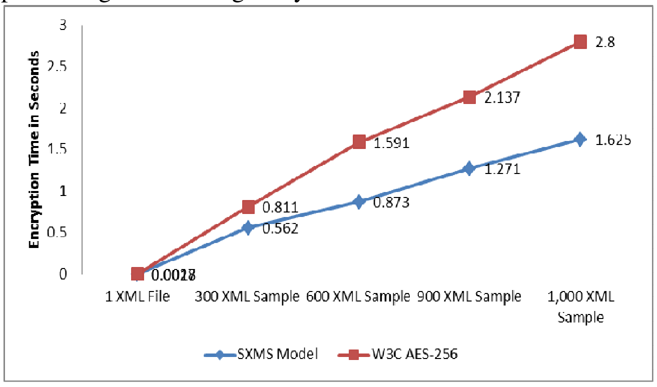


Figure 13 Comparisons chart between SXMS and W3C model using 256-bit

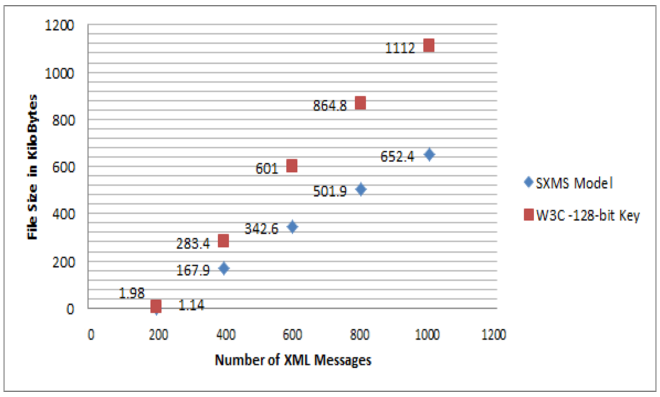


Figure 14 File size comparisons between SXMS and W3C using 256-bit key

Finally, figures 15, 16 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 measurable amount of performance improvement using SXMS model.

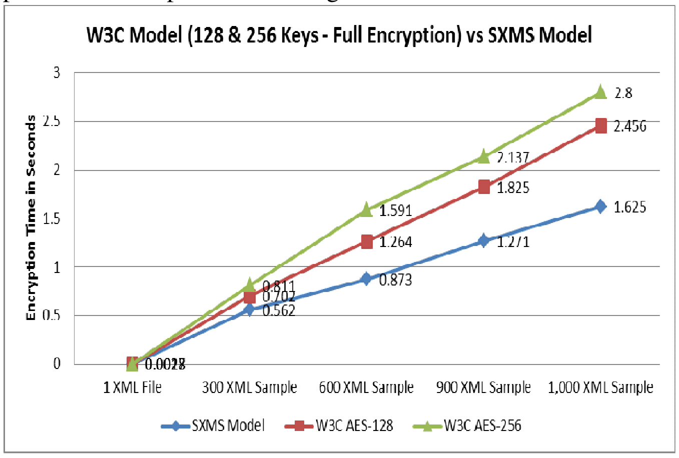


Figure 15 Performance comparisons between SXMS and XML using 256-bit

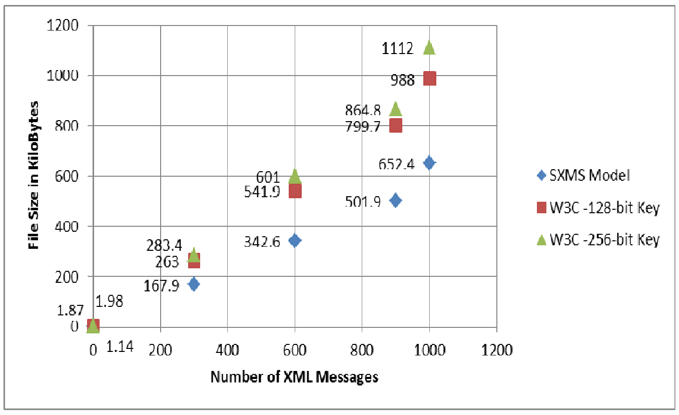


Figure 16 File Size comparisons between SXMS and XML using 256-bit key

Stage2: 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 9 represents the number of occurrences for transactions marked with "High" and "Medium" across the three layers.

TABLE 9: APPEARANCES FOR EACH CLASSIFICATION LAYER

Classification Layer	"High" Appearances	"Medium" Appearances	Percentage (High + Medium)
Layer1 (Account)	274	43	28.8%
Layer 2 (Details)	425	484	82.6%
Layer 3 (Environment)	299	457	68.7%

Table 10 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 10: PERFORMANCE EVALUATION FOR STAGE 2 – EXPERIMENT 1

Stage 2 Exp 1 (Partial Encryption)	**Processing Time**		**File Size**	
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	0.562 MS	0.578 MS	167.9 KB	244 KB
600 XML	0.873 MS	0.984 Sec	342.6 KB	510.2 KB
900 XML	1.271 Sec	1.422 Sec	501.9 KB	740.7 KB
1,500 XML	**1.963 Sec**	**2.218 Sec**	**810.1 KB**	**1203.6 KB**

As seen in Table 10, 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 a 11.4% improvement in processing time for the 1,500 messages.

Table 10 also illustrates files size reduction encrypting XML messages using SXMS model, table shows a measurable 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 measurable amount of space and bandwidth on large scale.

Figure 17 illustrates the comparison between SXMS model and W3C encryption standard using key size of 128-bit.

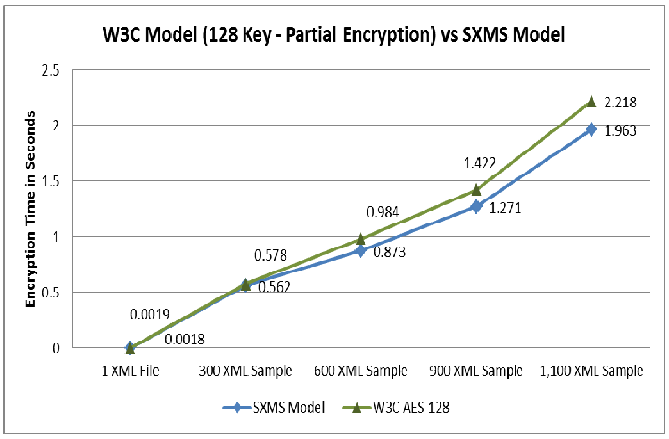


Figure 17 Performance comparisons between SXMS and W3C Standard using AES-as128 Key

Figure 18 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.

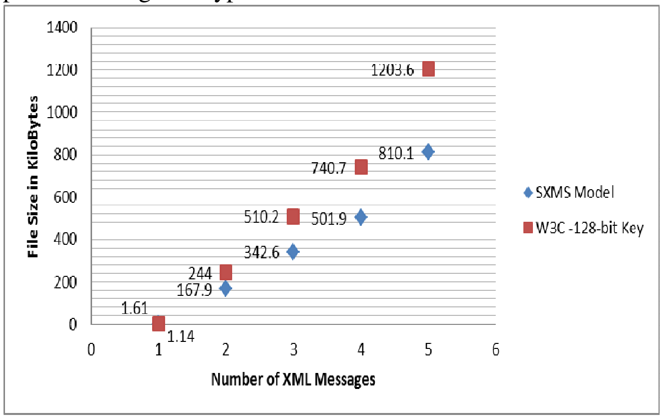


Figure 18 File size comparisons between SXMS and W3C Standard using AES-128 Key

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

TABLE 11: PERFORMANCE EVALUATION FOR STAGE 2 – EXPERIMENT 2

Stage 2 Exp 2 (Partial Encryption)	**Processing Time**		**File Size**	
Message Set	**SXMS Model**	**W3C 128 bit**	**SXMS Model**	**W3C 128 bit**
1 XML File	0.0018 MS	0.0021 MS	1.14 KB	1.72 KB
300 XML	0.562 MS	0.687 MS	167.9 KB	269 KB
600 XML	0.873 MS	1.42 Sec	342.6 KB	588.4 KB
900 XML	1.271 Sec	2.026 Sec	501.9 KB	813.9 KB
1,500 XML	**1.963 Sec**	**2.899 Sec**	**810.1 KB**	**1399.6 KB**

As seen in Table 11, 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 11 also illustrates files size reduction encrypting XML messages using SXMS model, table shows a measurable 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 measurable amount of space and bandwidth on large scale.

Figure 19 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.

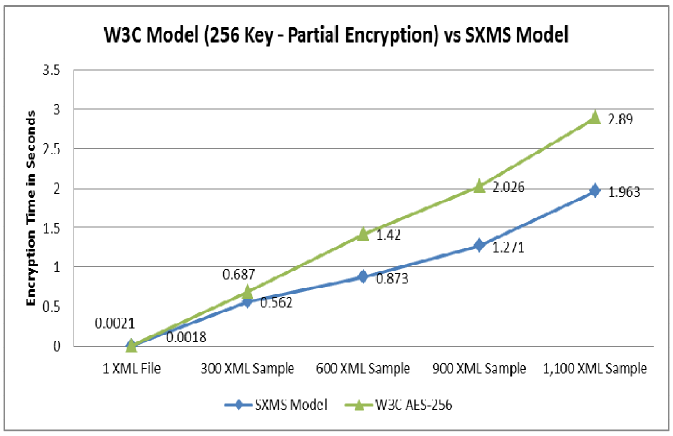


Figure 19 comparisons between SXMS and W3C Standard using AES-256

Figure 20 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.

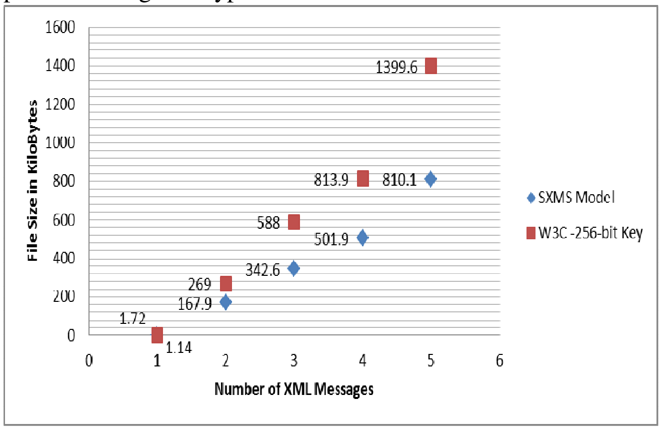


Figure 20 File size comparisons between SXMS and W3C model using 256bit

Finally, Figure 21, and Figure 22 illustrate performance improvements and file size reduction comparison between SXMS model and W3C model for both experiments in stage 2 showing a measurable amount of performance improvement and size reduction on a large scale using SXMS model.

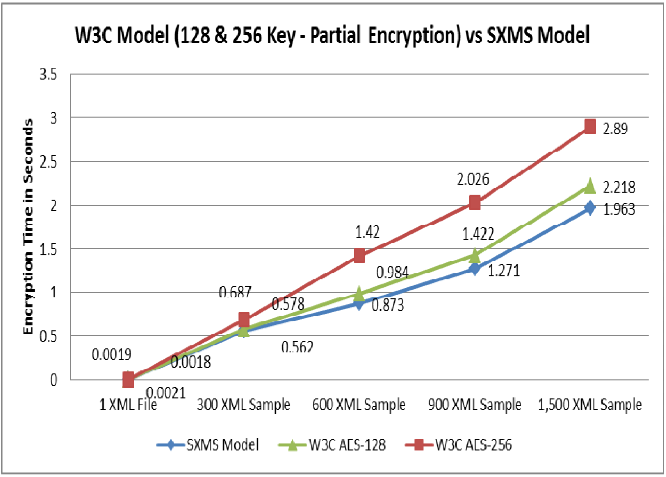


Figure 20 comparisons between SXMS and W3C Standard using different keys

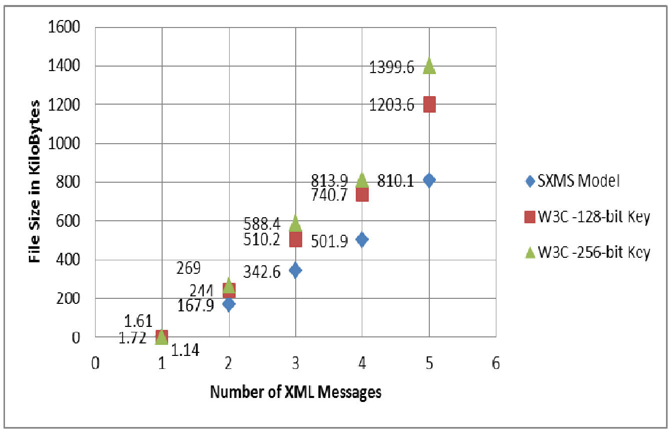


Figure 21 file size comparisons between SXMS and W3C Standard using different keys

V. Conclusion and Future Work

In this paper, a novel approach for securing financial XML messages using intelligent mining fuzzy classification techniques has been proposed.

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 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 classification processes based on Mamdani inference. 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 recommendation. Nodes that are assigned an importance level value of "High" will be encrypted using the AES encryption algorithm with a key size of 256 bit to ensure maximum security is performed. Nodes that are assigned an importance level value of "Medium" will be encrypted using the AES encryption algorithm with a key size of 128 bit. An implementation was performed on a real-life environment using online banking systems to demonstrate its flexibility, feasibility, and functionality. Our experimental results of the new model verified tangible enhancements in encryption efficiency, processing time reduction, and financial XML message utilization.

Each unit in our SXMS model acts independently as a separate system. Taking into consideration such flexible nature allows and motivates future work and enhancements. The following points describe the future work on each unit within our SXMS model:

• Fuzzy classification phase: We can utilize supervised machine learning techniques to automate the fuzzy rule generation process, in order to reduce the human expert knowledge intervention and increase performance of the phishing detection system. This can be achieved by generating classification rules using well known classifiers, for example we can use: PRISM [25], C4.5 Decision Tree [26], Ripper [27], k-nearest neighbor classification (kNN) [28], naïve bayes classification [29], linear least squares fit mapping [30], and the vector space method [31]. These mining association classification rules can be combined with fuzzy logic inference engine to provide efficient and competent techniques for importance level extraction.

• Encryption phase: We can utilize a different encryption scheme, asymmetric algorithms can be deployed. We have deployed symmetric encryption due to the efficiency and processing time outperforming asymmetric encryption algorithms. Even we can change the symmetric encryption algorithm to something different like DES, triple DES, and Blowfish. Researchers will be able to test and measure performance for any replaced encryption algorithm. Also usage of the encryption keys can be change to reflect different key size for each importance level assigned. For example we can assign an encryption key of size 192 bit instead of 256 bit for the importance level "High" value.

• We can create multiple instances of SXMS whereby it handles XML messages based on load balancer designed to distribute XML messages on multiple SXMS instances. By performing this distribution it will boost the processing speed 2x or even more depending on the new instances created and used. However, such initiative might be high cost on resources used.

REFERENCES

[1] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. W3C, Feb. 1998

[2] Fan, M. Stallaert, J. and Whinston, A. B.: The Internet and the Future of Financial Markets, Communications of the ACM, 43(11):83-88, November 2000.

[3] XML Encryption Syntax and Processing (W3C Recommendation), 2003.

[4] XML-Signature Syntax and Processing (W3C/IETF Recommendation), February-2002

[5] XML Key Management Specification (XKMS 2.0). http://www.w3.org/TR/2005/PR-xkms2-20050502/, 2 May 2005

[6] Shirasuna, S., Slominski, A., Fang, L., Gannon, D., 2004. Performance comparison of security mechanisms for grid services. In: Fifth IEEE/ ACM International Workshop on Grid Computing. IEEE Computer Society, pp. 360–364.

[7] Park, N., Kim, H., Chung, K., Sohn, S., Won, D., 2006. XMLsigncryption based lbs security protocol acceleration methods in mobile distributed computing. Computational Science and Its Applications – ICCSA 2006', vol. 3984. Springer, Berlin/Heidelberg, pp. 251–259.

[8] M. Liu, D. Chen and C. Wu. The continuity of Mamdani method. International Conference on Machine Learning and Cybernetics, Page(s): 1680 - 1682 vol.3, 2002.

[9] "Oasis security services (saml) tc," "http://www.oasis-open.org/committees/security/".

[10] Organization for the Advancement of Structured Information Standards (OASIS), Extensible Access Control Markup Language (XACML), V2.0, February 2005.

[11] ContentGuard. XrML: The digital rights language for trusted content and services. http://www.xrml.org/, 2001.

[12] SOA Approach to Integration, Packt Publising, M. Juric, P. Sarang, R. Loganathan, F. Jennings, 2007.

[13] Ed Simon. XML Encryption: Issues Regarding Attribute Values and Referenced, External Data. W3C XML-Encryption Minutes, March 2000. Session 3, Boston, MA.

[14] Imamura, T., Clark, A., Maruyama, H., 2002. A stream-based implementation of XML encryption. In: XMLSEC 2002: Proceedings of the 2002 ACM Workshop on XML security. ACM Press, pp. 11–17.

[15] Christian Geuer-Pollmann. XML Pool Encryption. ACMWorkshop on XML Security, November 2002. Institute for Data Communications Systems, University of Siegen.

[16] DIFFIE,W. ANDHELLMAN,M. E. 1976. New directions in cryptography. IEEE Trans. on Information Theory IT-22, 6 (Nov.), 644–654.

[17] Hwang, G.-H. & Chang, T.-K. 'An operational model and language support for securing xml documents.', Computers & Security 23(6), 498–529, 2004.

[18] L.A. Zadeh, Fuzzy Sets, Information and Control, 1965

[19] Mahant, N. (2004) Risk Assessment is Fuzzy Business – Fuzzy Logic provides the Way to Assess Off-site Risk from Industrial Installations, Bechtel, Australia.

[20] Ricardo Rosario. Secure XML An Overview of XML Encryption, November 2001

[21] Abhishek Gaurav , Reda Alhajj, Incorporating fuzziness in XML and mapping fuzzy relational data into fuzzy XML, Proceedings of the 2006 ACM symposium on Applied computing, April 23-27, 2006, Dijon, France

[22] Ma Z. Fuzzy XML data modeling with the UML and relational data models. Data & knowledge engineering. 2007;63:972-996.

[23] Tseng C. Universal fuzzy system representation with XML. Computer standards and interfaces. 2005;28:218-230.

[24] Fu Zhang, Z.M. Ma, Li Yan, Construction of fuzzy ontologies from fuzzy XML models, Knowledge-Based Systems, Volume 42, April 2013, Pages 20-39, ISSN 0950-7051, 10.1016/j.knosys.2012.12.015.

[25] Cendrowska, J. (1987) PRISM: An algorithm for inducing modular rules. International Journal of Man-Machine Studies. Vol. 27, No. 4, (pp.349-370).

[26] Quinlan, J. (1996) Improved use of continuous attributes in c4.5. Journal of Artificial Intelligence Research, Vol. 4, No. 1, (pp. 77-90).

[27] Cohen, W. (1995) Fast effective rule induction. Proceedings of the 12th International Conference on Machine Learning, (pp. 115-123). CA, USA.

[28] Guo G, Wang H, Bell D, Bi Y and Greer Y (2004): Using kNN Model for Automatic Text Categorization, Journal of Soft Computing, Springer-Verlag Heidelberg.

[29] McCallum, A. and Nigam, K. A Comparison of Event Models for Naive Bayes Text Classification. in AAAI-98 Workshop on Learning for Text Categorization, Madison,WI, 1998, 41-48.

[30] Yiming Yang and Christopher G. Chute. An application of least squares fit mapping to text information retrieval. In Proceedings of the ACM SIGIR, pages 281--290, Pittsburgh, PA, June 1993. 163

[31] Susan Gauch, Juan M. Madrid, Subhash Induri, Devanand Ravindran, and Sriram Chadlavada. KeyConcept: A Conceptual Search Engine, Information and Telecommunication Technology Center, Technical Report: ITTC-FY2004-TR-8646-37, University of Kansas.

Faisal T. Ammari (F.Ammari@hud.ac.uk) currently is a research student in the University of Huddersfield, UK. He is currently managing the business solutions department at Jordan Ahli Bank in Jordan since 2009. His main research interests are within web technologies and advanced security especially in financial sector.

Professor Zhongyu (Joan) Lu (joan.lu@hud.ac.uk) is in the Department of Informatics at the University of Huddersfield, UK. She was a Team Leader of IT Department in an industrial company before she jointed university. She has successfully conducted two industrial projects in the area of XML and database systems, collaborating with Beijing University, China, during her working in the company.

Maher Aburrous (maher198@hotmail.com) is an Assistant Professor in the Software Engineering Department, Faculty of Engineering & Applied Science at Al Hoson University - UAE, he has an intensive experience in Network Technology and Internet security, used to be an IT manager in the at Jordan Ahli Bank.