Improved Banking XML Transaction Encryption Using Tag Fuzzy Classification

Faisal T. Ammari, Joan Lu

Abstract: **In this paper we present a novel approach for securing financial XML transactions using intelligent fuzzy classification techniques. Given an XML message X, our approach defines the process of classifying XML content to assign a unique value, which indicates the data sensitivity declaring importance level for each XML tag. The classified message Xs includes this new modified attributes with importance level value assigned for each tag. The framework also defines the process of securing classified financial XML message by performing element-wise XML encryption on selected parts defined in Xs. Based on our approach, we define which encryption algorithm is more appropriate to be deployed on selected parts depending on importance level attribute defined in Xs.**

An implementation has been performed on a real life environment using online banking systems to demonstrate its flexibility, feasibility, and security. Our experimental results of the new model verified tangible enhancements in encryption efficiency, processing time reduction, and financial XML message utilization.

Index Terms— **Data Classification, XML Classification, XML Encryption, Fuzzification**

I. Introduction

XML [1] gained popularity being a leading standard in data exchange among various systems and applications. Flexibility, adaptability, and ability to use plain text to encode set of information gives XML messages the capability to be read and understood without any special reader or interpreter. Various businesses encouraged adopting this communication medium in their business model and functional communication messaging [2]. Increase demand on using XML in mission critical applications brought more attention on the security level surrounding XML structure and method of communication. Many recommendations have been proposed to secure outgoing XML messages. W3C first came up with XML Encryption [3], XML Signature [4], and XML Key Management [5]. These security models achieved targeted XML protection, however results deploying these models involved in performance issues, presentation problems, and high resources usage.

W3C first recommended XML as the standard for data representation over the web, and due to its flexible nature many businesses started to move their business communication messaging to XML, Eliminating the complexities of standard communication messaging.

Excessive use of XML in exchanging data among different business entities created an aligned interest securing data being transferred. Many models have been proposed to protect exchanged messages, on the network level [10, 11], and on the XML level. Among proposed models the World Wide Consortium (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].

XML Encryption standard defines how to encrypt the XML message, fully encrypting the entire message, partially 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, few issues rose concerning performance and inefficient memory usage [12, 13] which leaves a space for improvements. Symmetric or asymmetric encryption algorithms can be deployed. during encryption process data objects are encapsulated within a defined encryption element called *EncryptedData*, this element have essential sub elements describing how data is being encrypted, first sub-element is *EncryptionMethod* determining which encryption algorithm is being used within XML message. Second sub-element is *EncryptedKey* which is used to transport encryption keys between sender and receiver, it also can be used individually in a separate XML message. *ds:KeyInfo* is the third sub-element to specify the associated keying material. Another major element which is *CipherData* that is mandatory element providing the encrypted data, it must contain the encrypted octet sequence of base64 encoded text of the *CipherValue* element. Another way is by providing a reference to external location which contains encrypted octet sequence location via another element called *CipherReference.*

While XML encryption provides a way to encrypt specific parts in the XML message, fuzzy classification has been considered to facilitate information retrieval for only needed information from XML messages instead of retrieving all information included within each XML message, the process is performed by clustering included information into sub categories segmented in a pre-defined scheme. fuzzy classification can be achieved in three ways, first by using the textual content of the message and then use any standard text categorization technique to fetch the data, second by using only the structure of the XML message, and third by using a hybrid approach using both the textual content and the structure. Once XML messages have been classified we have the choice of assigning an importance level for the segmented tags within each XML message, importance level assigned is based on a set of rules defined by deploying Mamdani fuzzy inference [23].

II. Literature Review

Flexibility, expressiveness, and usability of XML formed a motive for researchers to shed more light on XML security. Researchers focused their interest in securing XML data due to the increased usage of XML in many business and educational cases. Efficient models have been proposed [3, 4, 5, 16, 17, 18] to add a secure layer over exchanged XML data. Models main purpose is to ensure data confidentiality and authenticity. Many XML threats [6] have been discussed to be taking under consideration like Oversized Payload, Schema Change, XML Routing, and Recursive Payload. Such threats forced researchers to pay more attention on securing exchanged XML messages.

World Wide Web Consortium (W3C) is working on XML security standards. The 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 possible to encrypt an element while keeping the descendants of the same node unchanged, and also it is impossible to handle attribute encryption. Therefore, a solution has been proposed [9] to handle this limitation, Ed Simon proposed to replace the attribute to be encrypted with *EncryptedDataManifest* attribute and include any other details inside the element. Another solution was proposed to use XSLT for attribute transformation into elements to perform the encryption process. However, suggested solution didn't face success as the decrypted parts are required to be transformed back to the original attributes for message validation against corresponding XML schema.

A stream-based parser for XML encryption [8] has been developed in 2002 between Apache and IBM. Their parser Xerces2 was built to act as a pipeline of different components. Configuration for XML encryption built as: XML --> Scanner --> Validator --> Parser --> API and then expanded as: XML --> Scanner --> Validator --> Encryptor --> Parser for the encryption phase. In case of decryption process they built it as: XML --> Scanner --> Decryptor --> Validator --> Parser --> API. Test cases and implementation achieved 0.7–26% reduction of processing time in the encryption process for files with sizes larger than 2KB and 34–88% in the decryption process for any file size. Best performance for the two phases (encryption, decryption) is achieved if the message size is in the range from 100 kB to 200 kB before the encryption process. However, Most of the implementation of XEnc uses DOM instead of SAX API. DOM is able to parse decrypted data efficiently and correctly. But DOM is known to have higher cost in time and space compared to SAX API as it parses XML message in memory.

A system has been proposed by [10] 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 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, 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 recipients' key before final submission. Although this model solves the issue of removing confidential material away from the main message and hides the size of encrypted content but it has disadvantages as following:

- 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 pool of node keys, a decent increase in message size is noticed.
- High resources usage and bandwidth allocation, more storage and more processing power is needed
- Unique node key has to be generated for each node.

III. System Model and Design

Our system design was built whereby each unit in system act as an independent unit, performing set of operations to deliver set of outputs required for the next following phase. Core of the system is dependent on two main modules, the first to perform XML classification using fuzzy logic techniques [7]. 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 mainly responsible for defining an attribute value and assign it to an existing XML tag named "ImportanceLevel", assigned value will be used to define the security level needed in the next phase.

A. Measurements of Success

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

1) XML Fuzzy classification: ability to classify XML messages by using a combination of system automation classification and human input classification to achieve highest credibility. A set of operations are performed on each XML message Xs:

- Flatten XML message, by flatten XML message we will be able to classify based on textual content

- XML content is categorized into three basic layers, each layer represent a state, each layer has a set of tags and their values
- Assign a value for the "Importance Level" tag which uses a value from (High, Medium, Low), tag exists in each layer header whereby tags included within layer inherited its value
- Following classification process, content are forwarded all based on its "Importance Level" value

2) XML Encryption: Element-wise XML encryption is performed on message Xs, Encryption algorithm is being used depending on assigned importance level attribute (high, medium, low). Multiple encryption algorithms can be used on the same message, our model uses AES encryption algorithm with two key values (128 bit, 256 bit) whereby AES with 128 bit key wrap is being performed on tags with "Medium" importance level value, AES with 256 bit key wrap is performed on tags with "High" importance level value. Tags with "Low" importance level are being forwarded as is without any type of encryption.

Fig. 3 illustrates system model and components involved forming our approach.

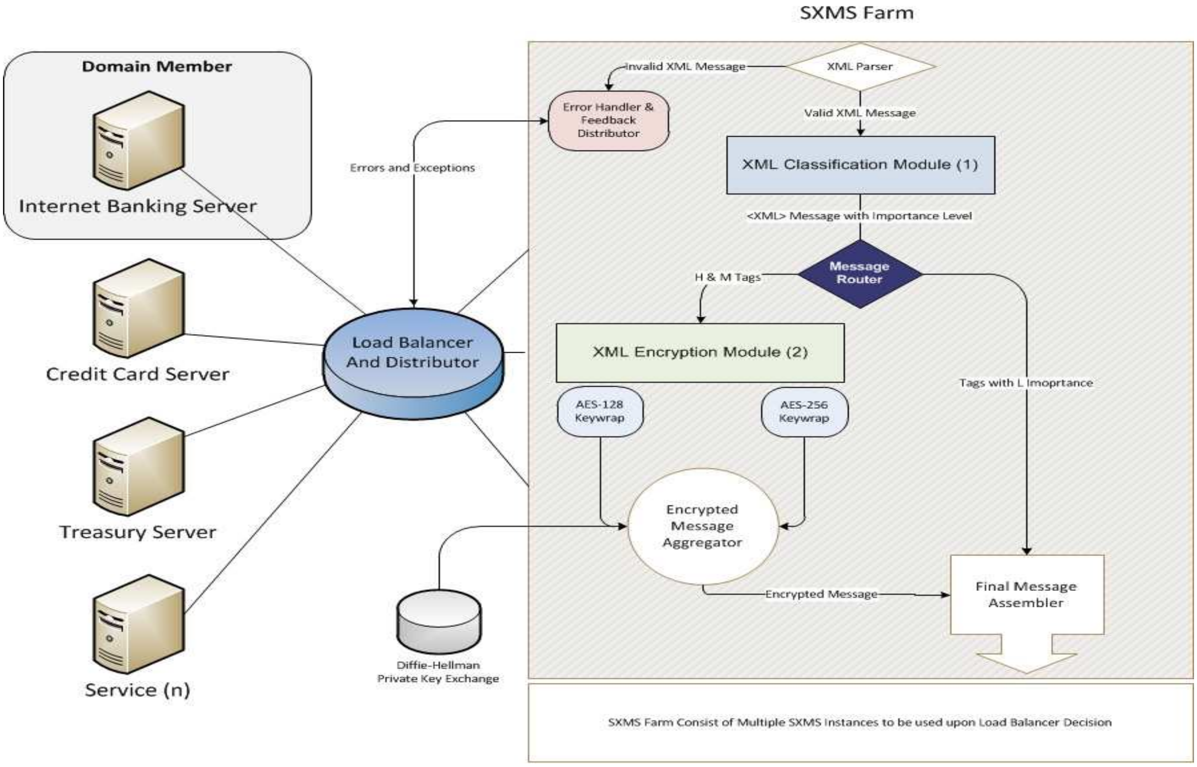

SXMS Farm
Domain Member
Internet Banking Server
Credit Card Server
Treasury Server
Service (n)
Load Balancer And Distributor
Errors and Exceptions
Invalid XML Message
XML Parser
Error Handler & Feedback Distributor
Valid XML Message
XML Classification Module (1)
<XML> Message with Importance Level
H & M Tags
Message Router
XML Encryption Module (2)
Tags with L Imoprtance
AES-128 Keywrap
AES-256 Keywrap
Encrypted Message Aggregator
Encrypted Message
Final Message Assembler
Diffie-Hellman Private Key Exchange
SXMS Farm Consist of Multiple SXMS Instances to be used upon Load Balancer Decision

Fig. 3 Main System Design

B. System Design

System core has been built based on two major phases, each phase has discrete scope acting as an independent unit and also as a part of the whole system. Phase one involve performing set of fuzzy classification techniques on XML messages, fuzzy classification process is mainly for deciding similarity of different standards within 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 "ImportanctLevel" to the mentioned tag so we can use it as identifier for the next phase. Phase two involve applying element-wise encryption to different parts with each XML message, Encryption could be for the whole message, some elements, and some attributes of an element of an XML message. ImportanceLevel value assigned in phase one is used to decide which type of the encryption is performed, also decides which parts of the XML message to be encrypted. We base our encryption on W3C recommendation [3].

1. Fuzzy Classification Module

This phase perform a set of intelligent techniques to assign a new value which is the importance level for each XML tag, main idea is distinguish which parts of the message to be encrypted using AES-128 bit key encryption and which to be encrypted using AES-256 bit key, usage of the key depending on the importance level value (high, medium, low), whereby we deploy 128 bit key on tags with "medium" ImportanceLevel and 256 bit key on "high" ImportanceLevel value. Tags with "low" importance level value are forwarded directly to message assembler where no encryption is being performed. Phase is using fuzzification techniques of a set of input variables based on 10 characteristics extracted from the XML message, all depending on previous knowledge experience and expertise backgrounds. 10 characters are defined as following:

1) Transaction Amount: Financial institutions set a pre-defined transaction limits, limits allow users to perform transactions with specified limits on daily basis. Range of transaction limits are defined based on local policy within each institution.

2) Transaction Currency: A well-defined list of allowed currencies that can be used in any online or offline system. Each currency has its own set of risk variables depending on usage and importance.

3) Account Type: Accounts are segmented within each institution, segmentation is being performed to be able to apply set of internal rules on selected segments.

4) Transaction Notes: Exceptions are being placed upon unusual activity on a specific account, such exceptions will raise a flag in any transaction being processed to handle the exception before process is completed.

5) Profile ID: A unique identifier for the destination account owner, Value is being set during system integration and profile creation process.

6) Account Tries: How many times the account is being used in the system, the more usage means more trust whereby history of the account is known and trusted.

7) Incorrect Password Tries: Number of times users tried to enter the password incorrectly to complete the financial transaction.

8) Time Spent on Service: Time spent navigating the service before performing the transaction, time range is set based on bank's policy taking into consideration peak hours.

9) Daily Transactions: How many transactions are performed before doing the financial transaction.

10) Transaction Time: 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 historical transactions range.

2. Fuzzy Methodology

Our fuzzy classification phase is based on Mamdani [23] fuzzy inference, performing the basic four steps in Fig. 4.

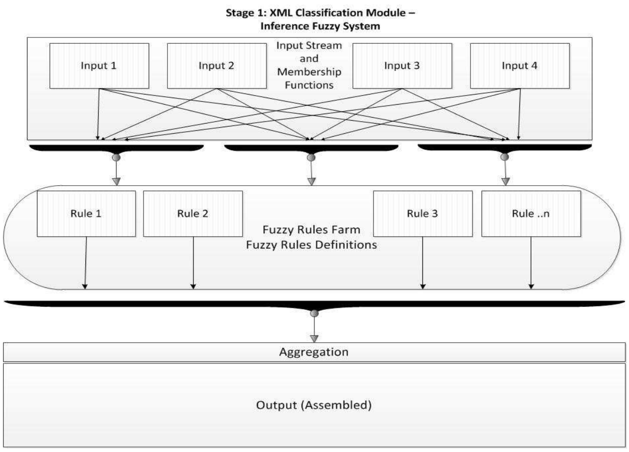


Fig. 4 Mamdani fuzzy inference system

Step 1 (Fuzzification): Taking the crisp input X, Input Y, and determines the degree to which of these inputs belong to and where to fit in the fuzzy set.

Step 2 (Rule Evaluation): Taking the fuzzy inputs and apply to the qualified fuzzy rules. fuzzy operators (AND / OR) are being used in case of any uncertainty to get a single value, 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) Output should have a clear crisp value where it will be assigned to each tag classified.

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

Medium: Tag is somehow important, tag attribute is assigned the value of medium so an encryption will be applied using AES algorithm with key of 128 bit.

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

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 has its own weight and criteria. Layers are Account Layer, Details Layer, and Environment Layer, Fig. 7 represent layers distribution.

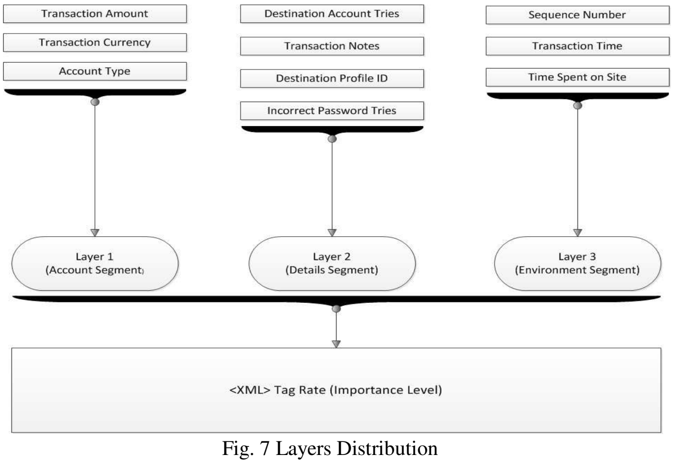


Fig. 7 Layers Distribution

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, rule is based on "IF-THEN" rule, rule base should contains a number of entries depending on how many layer members exist. For example layer 1 has three members and we have three output expected so the entries should be calculated as (3^3) = 27 entries presenting the rules for that layer. Table 1 represents a sample of rule base for layer1 (Account Layer).

TABLE 1: RULE BASE SAMPLE FOR LAYER 1

Amount	Currency	Type	Account Layer
Non-Sensitive	Sensitive	Non-Sensitive	**Medium**
Non-Sensitive	Sensitive	Non-Sensitive	**Medium**
Non-Sensitive	Normal	Sensitive	**High**
Non-Sensitive	Normal	Sensitive	**High**
Non-Sensitive	Normal	Normal	**Medium**

Final evaluation is depending on finding centre of gravity as 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 importance level attribute defined and assigned.

4. Encryption Module

Encryption phase has two possibilities: first one is to perform an element-wise encryption using AES algorithm with key size of 256 bit, second is to perform an element-wise encryption using AES algorithm with key size of 128 bit. Key size is determined by the ImportanceLevel value assigned in fuzzy classification phase. Fig. 10 illustrates the process of encryption. Tags with "Low" ImportanceLevel will be forwarded directly to message composition stage without performing any type of encryption.

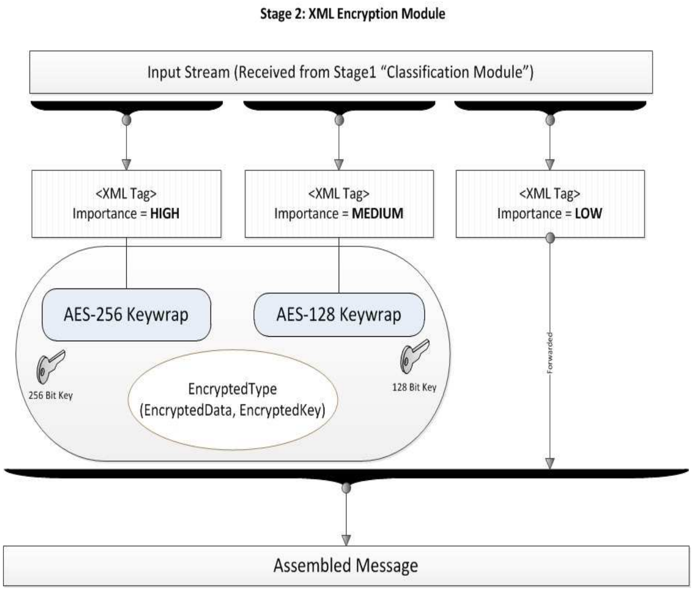


Fig. 10 Encryption module layout

System flexibility allows us to change encryption standard upon convenience, as stated encryption phase act as an independent unit whereby it can operate separately. Such rich flexibility can grant us to change encryption standard used, DES or triple DES encryption can be deployed easily whereby we only need to place the new algorithm class into phase core.

Tags related to the parent tag are encrypted as well using the same level of encryption. Child tags behavior is taken from the parent "ImportanceLevel" value. In Fig. 11 (Account Holder, Account Number, Amount, Currency, and Type) tags are encrypted using AES encryption with 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, categorization process in our model is built based on relevance and parent tag evaluation.

5. Message Utilization

To ensure maximum efficiency, message utilization is being calculated by subtracting the actual message size from the potential output divided by the potential output as following:

$$OG = \frac{ActualOutput - PotentialOutput}{PotentialOutput} * 100$$

Sample Utilization:
Original Message Size = **15k**
Classified Message: **4k / 7k / 4k**

$$OG = \frac{15 - 11}{11} * 100 = 36.6\%$$

In above example message has been utilized to achieve **36.6%** out of original message processing time.

Once utilization process is completed, Messages are ready for final submission to selected destination. However keys used during encryption process should be transferred to decryptor in destination using a secure and private way. We use Diffie-Hellman [24] key exchange for keys hand over between source and destination.

IV. EXPERIMENT AND RESULTS

We have developed two implementations of the model. The first implementation is using the internet banking service provided in one of the leading banks in Jordan. 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.

First implementation conducted using a sample of 1,000 records, Sample records 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 system design. Table 2, Table 3, and Table 4 illustrate sample of data provided 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**
Non-Sensitive	Non-Sensitive	Normal	**Low**
Normal	Normal	Sensitive	**Medium**
Normal	Normal	Normal	**Medium**
Sensitive	Non-Sensitive	Sensitive	**High**

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

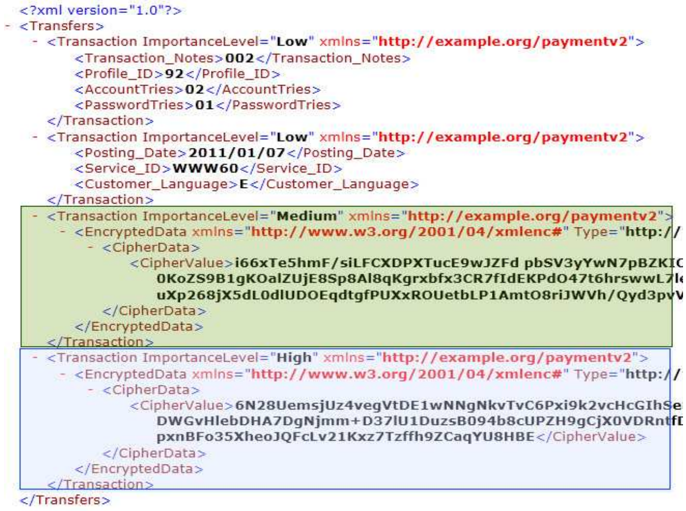
TABLE 4: SAMPLE OF DATA RECEIVED CLASSIFIED FOR LAYER 3

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

Our model main goal is to optimize and increase encryption processing time, therefore we tried to fetch number of records that have an overall “High” ImportanceLevel.

Second implementation conducted using a sample of 1,200 records, Sample records selected randomly presenting various transaction types like external transfers for a period of five months different than the first implementation period. System classified sample messages into three layers depending on 10 characteristics described in system design.

Assigned importance level is used as an indicator for which type of encryption is needed on corresponding node. Fig. 16 illustrates how the XML message looks like after encryption phase performed on selected parts with "High" and "Medium" ImportanceLevel value. As seen in Fig. 16 only selected parts are being encrypted whereby two different keys are deployed, first is AES-128 bit on tags marked with “Medium” ImportanceLevel, second is AES-256 bit on tags marked with “High” ImportanceLevel.



```
  <?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">
    - <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#" Type="http://
      - <CipherData>
          <CipherValue>i66xTe5hmF/siLFCXDPXTucE9wJZFd pbSV3yYwN7pBZKIC
            0KoZS9B1gKOalZUjE8Sp8Al8qKgrxbfx3CR7fIdEKPdO47t6hrswwL7le
            uXp268jX5dL0dlUDOEqdtgfPUXxROUetbLP1AmtO8riJWVh/Qyd3pvV
        </CipherData>
      </EncryptedData>
    </Transaction>
  - <Transaction ImportanceLevel="High" xmlns="http://example.org/paymentv2">
    - <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#" Type="http://
      - <CipherData>
          <CipherValue>6N28UemsjUz4vegVtDE1wNNgNkvTvC6Pxi9k2vcHcGIhSe
            DWGvHlebDHA7DgNjmm+D37lU1DuzsB094b8cUPZH9gCjX0VDRntfD
            pxnBFo35XheoJQFcLv21Kxz7Tzffh9ZCaqYU8HBE</CipherValue>
        </CipherData>
      </EncryptedData>
    </Transaction>
</Transfers>
```

Fig. 16 Encrypted XML message taken from first implementation

V. PERFORMANCE EVALUATION

Conducted experiments were made to evaluate and compare the performance of securing XML messages against W3C XML Encryption [3], using our intelligent fuzzy classification techniques we were able to secure XML messages in less processing time and more efficient way. Efficiency came from deploying encryption only on needed parts within XML messages, this allowed us to utilize each message before final submission to receiver.

Testing against W3C Encryption conducted in two sets, First set is by performing a full encryption on XML messages and measure the results against ours, we used symmetric AES encryption based on two key values (128 bit, 256 bit). Second set is by performing partial encryption using AES encryption with two key values (128 bit, 256 bit) to measure it against ours as well. First set experiments conducted 20 times on the first sample of 1,000 records. Table 7 illustrates time needed for each key and model to perform the encryption process.

TABLE 7: STAGE 1 EXPERIMENT AND RESULTS

Model	Processing Time (per file)	Per 1,000 files
W3C Full Encryption - **AES128**	**0.0623 MS**	62 sec
W3C Full Encryption - **AES256**	**0.0715 MS**	71 sec
Our Model - Classified (**Mixed Keys**)	**0.0349 MS**	**34 sec**

As shown in table 7 and taking the average of 20 tries, our model were able to achieve an improvement in processing time. First attempt we have conducted W3C AES encryption with key of size 128 bit to process the whole message, a sample of 1,000 XML messages have been used. Each XML file consist of a complete internet banking transaction, Number of tags for each transaction consist of 4 main tags presenting the whole transaction. We have processed the same sample of 1,000 records but using our model "which uses combination of keys depending on ImportanceLevel assigned".

W3C Encryption using 128 bit key size processed the 1,000 files in 62 seconds whereby our model achieved 34 seconds to encrypt the same sample, results reflect 45.1% improvement in processing time for the 1,000 records. Fig. 17 illustrates the comparison between the two models.

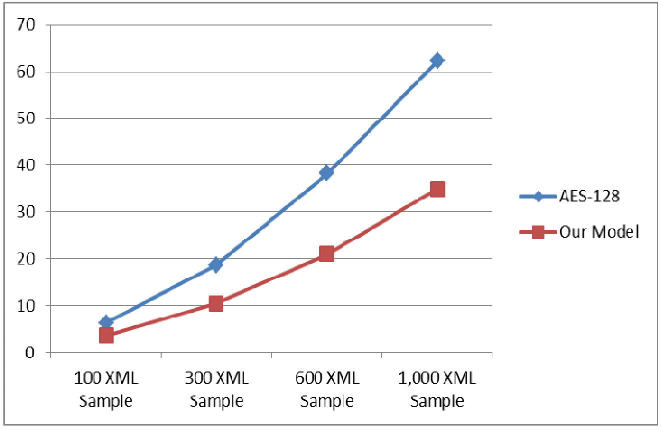


Fig. 17 Results of encrypting sample messages using full encryption

Second attempt we have conducted W3C AES encryption with key of size 256 bit to process the whole message, the same sample of 1,000 XML messages have been used. We have processed the same sample of 1,000 records but using our model. W3C Encryption using 256 bit key size processed the 1,000 files in 71 seconds whereby our model achieved 34 seconds to encrypt the same sample, Results reflect 52.1% improvement in processing time for the 1,000 records. Fig .18, 19 demonstrate the comparison between our model and W3C XML encryption using 256 bit key size and final comparison between the three experiments and processing time required for each model. The three experiments were conducted for the first set which uses full W3C XML encryption using two keys (128, 256 bit). Second set is performed using same sample but with deploying partial XML encryption against our model.

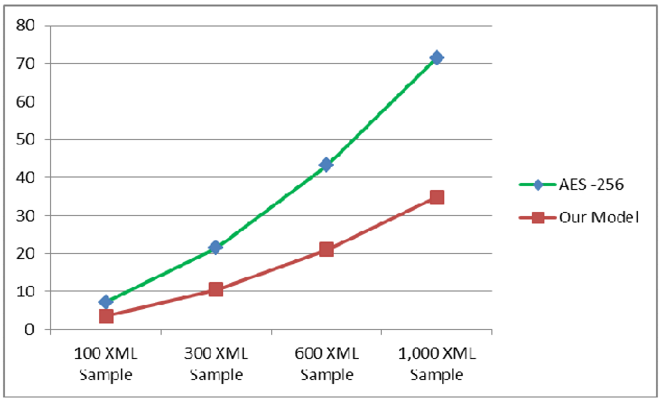


Fig. 18 Results of encrypting sample messages using full encryption

In second set we managed to encrypt the second sample of 1,200 records in 39.5 seconds (0.0395 MS per file), Our model encrypts the files using 128 bit key size if the importance level attribute is set to medium, 256 bit key size is used if the importance level attribute is set to high. Tags with no encryption just forwarded to message assembler for final message composition alongside encrypted parts. This approach gave us the edge to reduce processing time for each file being encrypted assuring only sensitive parts to be encrypted instead of the whole message. We deployed W3C Encryption using AES algorithm with key of size 128 bit to process pre-selected tags, selected tags are selected based on historical value assigned by experts within IT Department. Using W3C XML Encryption the second sample of 1,200 records processed in 45.8 seconds (0.0458 MS per file). Results reflect 13.7% improvement in processing time for the second sample of 1,200 records.

Second experiment in second set, we have conducted W3C AES encryption with key of size 256 bit to perform partial encryption on the message, the same sample of 1,200 XML messages have been used. W3C Encryption using 256 bit key size processed the 1,200 files in 59.2 seconds whereby our model achieved 39.5 seconds to encrypt the same sample, Results reflect 33.2% improvement in processing time for the 1,200 records. Fig. 20 demonstrates the comparison between our model and W3C XML encryption using 128, 256 bit keys.

TABLE 8: STAGE 2 EXPERIMENT AND RESULTS

Model	Processing Time (per file)	Per 1,000 files
W3C Partial Encryption - **AES128**	**0.0458 MS**	46 sec
W3C Partial Encryption - **AES256**	**0.0592 MS**	59 sec
Our Model - Classified (**Mixed Keys**)	**0.0395 MS**	**39 sec**

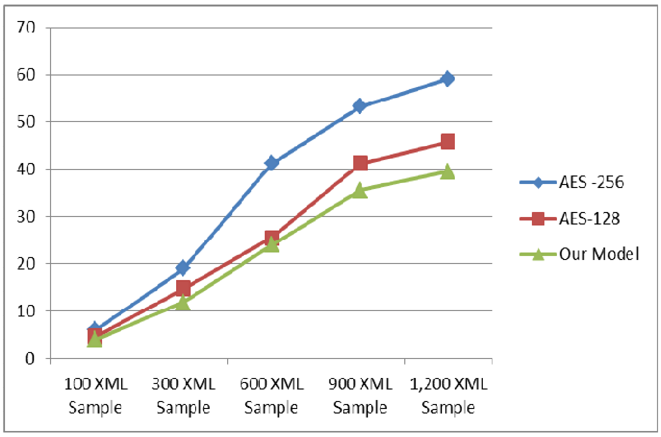


Fig. 20 Results of encrypting second sample messages by partial encryption

Message utilization is one of the steps performed before the final submission, Reason is to achieve maximum efficiency and improved performance. We calculate message utilization by the following:

$$\text{Output Gap} = \frac{\text{Actual Output} - \text{Potential Output}}{\text{Potential Output}} * 100\%$$

In our first set of experiments, we have managed to utilize 104.8% of the messages encrypted using W3C XML Encryption with key size of 256 bit as following:

Experiment 1 (W3C Full Encryption using 256k key):

In the same first set of experiments, we have managed to utilize 78.5% of the messages encrypted using W3C XML Encryption with key size of 128 bit as following:

Experiment 2 (W3C Full Encryption using 128k key):

In our second set of experiments, we have managed to utilize 49.8% of the messages encrypted using W3C XML Encryption with key size of 256 bit as following:

Experiment 3 (W3C Partial Encryption using 256k key):

In the same second set of experiments, we have managed to utilize 15.9% of the messages encrypted using W3C XML Encryption with key size of 128 bit as following:

Experiment 4 (W3C Partial Encryption using 128k key):

VI. 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 data sensitivity level within each XML message to find a degree of sensitivity for each tag in the message. Mining fuzzy classification process allowed us to assign a value to a new attribute added to the parent XML nodes. A value is determined upon applying set of classification processes based on Mamdani inference [23]. New value has been used to determine which type of encryption algorithm is being performed on selected tags, allowing us to secure only needed parts within each message rather than encrypting the whole message. XML encryption is based on W3C XML recommendation. Nodes assigned an importance level value of "High" will be encrypted using AES encryption algorithm with key size of 256 bit to ensure maximum security is performed. Nodes assigned an importance level value of "Medium" will be encrypted using AES encryption algorithm with key size of 128 bit. An implementation has been 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.

There are many directions for future work. First, we can use different mining classification and different encryption algorithms for the same framework. Also future enhancements can be done by enhancing the existing classification module by using a mixture of associative fuzzy classification rules that can work in combination with fuzzy inference system.

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

[7] 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.

[8] 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.

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

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