Journal of Computing and Information Technology - CIT 21, 2013, 2, 71–84
doi:10.2498/cit.1002135

Specifying Access Policies for Secure Content Dissemination of XML: A Technique Inspired by DNA Cryptography

Rajni Mohana and Deepak Dahiya

CSE Department, Jaypee University of Information Technology, Wakhnaghat, Solan, India

SOA helps to provide business agility by configuring entities to maximize loose coupling and reuse. XML is the most relevant means to provide interoperatablity among various entities. When in network, a XML file can be prone to hacking and unauthorized access, thus data integrity and confidentiality are the important issues of communication. Secure dissemination of an XML file is one of the techniques to ensure data integrity and confidentiality. This paper presents a secure dissemination technique such that extraneous data not meant for a legitimate consumer is inaccessible, there will be no information leak. The technique applies DNA cryptography due to its feature of compactness and simplicity. The technique encrypts the data and hides it in a garbage file; such that only legitimate consumer can see only the subscribed amount of data according to the access policies using the restriction enzymes. The paper also presents multicast dissemination interface that implements the proposed technique at the server level. The interface is built dynamically and asynchronously using a publish–subscribe methodology. The results indicate that the proposed technique not only satisfies the requirement specification of secure dissemination, but also points out its robustness in terms of time required to break the key. The technique is computationally secure as the time to crack the key is quite long and increases with increase in key length.

Keywords: secure dissemination, XML, DNA cryptography, restriction enzymes

1. Introduction

Service Oriented Architecture (SOA) [1] is defined as a deployment infrastructure on which new applications can be built quickly and easily. It helps to provide business agility in a Business to Business (B2B)/ Business to Consumer (B2C) applications by configuring entities (services, registries, contracts, and proxies) to maximize loose coupling and reuse[2]. Interoperatablity is a major issue in SOA as one has to tie heterogeneous business systems. Extensible Markup Language (XML) is the most relevant means to provide interoperatability [3]. XML structure has a tree structure nested to any level and depth. As per the definition given below, XML consists of an outer tree which can contain inner subtrees.

Def 1: A tree is formally defined as a finite set T of one or more nodes such that

1. There is one designated node called the root node of the tree, root (T)
2. The remaining nodes are partitioned into $m \geq 0$ disjoint sets $T_1, T_2, \ldots, T_m$, and each of these sets in turn is a tree. $T_1, T_2, \ldots, T_m$ are called as subtrees of the root. [4]

To simplify, An XML is a text file transported through network to enable communication among various webservices constructed over heterogeneous environment. When in network, it will be prone to hacking and unauthorized access. The integrity and confidentiality needs to be preserved as it may carry vital business information. Hence it becomes mandatory to find a solution for both out-house and in-house intrusions, such that transmitting data through network channels becomes secure. Security of the data is an important issue as having better QOS (Quality of Service) [5] is all in vain if data integrity cannot be treasured. The requirement specification is to design models and mechanisms (encoding of data) to avoid out-house intrusion, as well as in-house intrusions (by specifying and enforcing access control policies for secure dissemination of XML documents) [2].

In this paper we propose a technique to encrypt the data and an algorithm for secure dissemination of content in inter-enterprise and intra-enterprise networks. This novel technique explores the tree structure of XML and specifies the access policies of the various consumer such that only that part of data which is required by the consumer is visible to it, thus providing data integrity, privacy and access control.

1.1. Motivation

Consider a scenario where large documents at the source end called producer can be accessed by large number of service requesters called consumers as shown in Figure 1. Even a legitimate consumer can exploit the knowledge of context from the data elements it has access to [2]. The consumer should be able to view only relevant data which he has been authorized to view. Extraneous data not meant for a consumer should be inaccessible as flow of extraneous data may leak information, even when the data is encrypted. These access policies should be specified by the producer. In particular, the extraneous data is prone to off-line dictionary attacks even by legitimate consumer that can exploit contextual knowledge from the data elements it has access to [4]. Controlling the access through policies will improve sense of security inside the system, thus resisting to in-house attacks. The data is only visible to those consumers who have requested for it and are authorized to access it. Any secure dissemination approach should not only take care of data integrity, but also control the access to the data by the consumer. The approach should be scalable, efficient and able to handle many consumers for a single data. This problem gets complicated as the hierarchical organization of the content, different confidentiality and integrity requirements may exist for different portions of the same content [3].

The n-ary tree T represents the tree structure of a XML generated by the producer or the service provider. The access policy for the scenario shown in Figure 1.

Compactness, simplicity in implementation and time required to decode the data without having the right information is only the basic criteria for choosing an algorithm. Our proposal is inspired by DNA cryptography [8] as it provides high complexity at decoding end without right information.

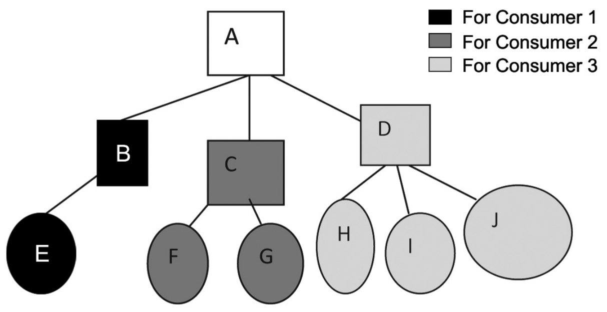


Figure 1. Tree structure of an XML data.

The XML data will be encrypted in a DNA strand. We describe a method of publish/subscribe approach where a user can subscribe for a data. The publisher will assign a Starting Restriction Enzyme (SRE) and Ending Restriction Enzyme (ERE). Restriction enzymes are molecular scissors that cut DNA into fragments at specific sites in their sequence.

The restriction enzymes will act as a flag to identify the subscriber of the message. Restriction enzymes will be added as a predecessor and successor of data. The dissemination interface routes the mapped XML content to the user whenever there is a new document published. The interface can check out that if the data is meant for a particular consumer, then it will route only that portion of data which will have the starting and ending restriction enzymes as mentioned by the subscriber or consumer.

The main contribution of the paper can be summarized as follows:

1. Securing the data in a DNA strand, thus securing the data and providing data integrity.
2. Specifying access policies of various consumers using the restriction enzymes.
3. Designing an interface for routing that can be used in non trusted domains for dissemination of sensitive content.

1.2. Outline of the Paper

The remainder of this paper is organized as follows: Section 2 initially provides a brief overview of the background knowledge required while Section 3 discusses the related study and the working of the secure dissemination technique is discussed in Section 4; further, Section 5 presents the algorithm and Section 6 presents an interface that implements the proposed technique. Section 7 presents a discussion of the technique based on various aspects; finally, Sections 8 and 9 summarize the conclusion and the list of references.

2. Background

The background knowledge contains information about various technologies used and details of their features which have been used.

2.1. XML

XML is a declarative and narrative language and needs parser of type DOM (The Document Type Object) to fetch out useful information described therein at processing end. One of the sequential techniques and a compact representation of tree structure is to represent tree in memory using linear list [3]. It is the most common representation of the tree states that the nodes can be stored by consecutive addressing. The tree in Figure 2 can be represented in the memory as (A (B (E), C (F, G), D (H, I, J)).

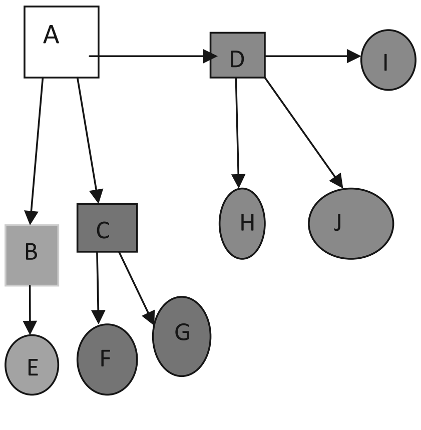


Figure 2. Linear list representation of the tree in memory.

The root A will have a link to all of its children from left to right; the children will further link to their children. Thus, if the producer wants to transmit the data which the consumer 3 has subscribed for, only the address of D should be known to the consumer and the whole subtree can be traced easily. Thus, any subtree is accessible if a consumer is able to locate the address of the root of the subtree. We are trying to exploit the above property of the tree, for securely disseminating the tree.

To ensure security of data, the linear list can further be encoded in a DNA strand. The properties of DNA are its compact nature and simplicity in implementation which makes it an ideal choice to encrypt the tree structure of XML data. DNA also provides high complexity at decoding end without right information.

2.2. DNA Cryptography

DNA cryptography has been proposed by Gehani et al. [10], Kartalopoulos [11] and Tanaka et al. [12] as a new born cryptography field. It's also called as DNA stenography, where we tend to hide data in a DNA strand with compactness and simplicity.

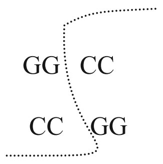
As we know, the DNA strand is composed of (A+C+T+G)*, thus possible patterns for this encoding format are $4! = 24$. According to Watson-Crick complementary rule [19] nucleotide base A is complement to T and C is complement to G. Take DNA digital coding into account, it should reflect the biological characteristics of 4 nucleotide bases, the complementary rule that $(\sim 0) = 1$, and $(\sim 1 = 0)$ is proposed in this DNA digital coding. According to the complementary rule, the complement of 0(00) is 3(11) and 1(01) is 2(10) and vice versa.

So among these 24 patterns, only 8 kinds of patterns (0123/CTAG, 0123/CATG, 0123/GTAC, 0123/GATC, 0123/TCGA, 0123/TGCA, 0123/ACGT, and 0123/AGCT) which are topologically identical fit the complementary rule of the nucleotide bases [19].

2.3. Restriction Enzymes

Restriction enzymes are molecular scissors that cut DNA into fragments at specific sites in their sequence. The enzymes degrade the foreign DNA by cutting the area that contains specific sequences of nucleotides. Since discovering the function of these enzymes, molecular biologists have isolated them from a variety of single-celled organisms for cutting DNA into fragments [13].

HaeIII, isolated from Haemophilus aegyptius, is an example of a restriction enzyme. Its recognition sequence is:



Wherever in the DNA strand, it will find the sequence shown above and it will cut the strand between G and C.

There is a large set of database of available restriction enzymes [16, 17]. The RBASE database registers 4000 restriction enzymes. We are not focusing on the scissor property of the enzyme, but are concerned about their property of identifying the recognition site as shown in above example. This property of the restriction enzyme is used to specify the access policy. The novelty lies in the scheme of specifying the access policies of various consumers, which is a computationally secure method, as proved in the Section 6.

The idea is to assign a SRE and ERE to a specific consumer so that when a data is prefixed and suffixed by the respective enzyme, the corresponding consumer can be located.

We have encountered that in the real world there are 4000 types of restriction enzymes, this will limit the number of combinations to 4000(SRE) * 4000(ERE). Any normal computer can easily crack the right combination of SRE and ERE as it would take at max 16000000 iterations. Hence we have not picked up real restriction enzymes, but our SRE and ERE consists of the sequence of bits {A,T,C,G} of length n, where $n \in \{1, 2, 3, \ldots\}$ making it a variable length key.

3. Related Study

Earlier researchers [2, 4], implemented secure dissemination using structural based routing. The routing model presented by them is based on multi-casting of document portions from an intermediate router to the subscribers. Essentially, the router may send the same document portion multiple times to the subscriber. In [21] a centralized publish/subscribe middleware is presented which is able to perform selective XML content delivery based on a shared ontology. It suggests semantic queries over domains concepts to compute a set of candidate concepts and over publisher's policies to check evolving policies for a subscriber the requestor asks for a set of concepts and therefore requires knowledge of the ontological structure. In [22-26] the work as a request response paradigm in client-server architecture is presented. The paper [27] integrates the XACML attribute model with OWL ontology and describes practical privacy filtering. The application is able to filter out information from XML documents, according to a set of XACML semantic privacy policies. It highlighted the need of a set of XACML facilities to evaluate decisions on hierarchic resources.

4. Working of the Proposed Secure Dissemination Technique

The dissemination follows a multicast based approach. A multicast tree exists in real network, particularly the internet. Reliability, security and congestion control are the bottlenecks for designing an efficient multicast topology-based protocol [15]. Our interface takes care of the security by ensuring encoding and secure dissemination as explained in Sections 3 and 4.

The interface ensures secure dissemination of sub trees of the XML data by using an interface for routing. A structure-based routing strategy builds and maintains a routing structure, such as a spanning tree, a routing table, or one or multiple paths, but it's static. If a change is required, it has to be changed in the entire router in the network. Hence we are implementing the dissemination through an interface as shown in Figure 3; the interface will encrypt the data and transmit it to all the consumers subscribed for the data. If any change is required, it can be implemented at the application level i.e., the interface, but not at the router's level. The administrator at the server end should mention the access policies of the consumer depending on the trust it has, it will then customize at the server end which user can access which part of data. Thus the administrator will handle secure communication and load balance. The technique consists of three things, three actors namely:

- Producer, who prepares an encrypted XML document to transmit through the network
- *N* number of consumer who has to read the message from the XML document
- An interface, which acts as disseminator which customizes the access policies for each customer by assigning a flag to the customer, such that the customer can extract the message meant for him through the XML file and decode it to get the message.

The secure dissemination technique encompasses the following procedure:

At the producer end the procedural flow involves:

- It prepares a temporary XML file to be transported to the consumer by prefixing the data with a tag called 'for consumer _“_.
- Later, the temporary file is picked and wherever a 'for' tag file is located, it's replaced by SRE and the data is suffixed by ERE of the concerned consumer number.
- The producer encrypts the content of all the XML elements in a DNA strand.
- Later, it transports it to various consumers.

At the consumer end when the data is received, the procedural flow involves:

- The consumer receives the encrypted file
- Later, it searches for the data that will fall between its specifies restriction enzymes.
- Consumer obtains data access to Root node and children address
- Data of the child nodes can be extracted from the addresses obtained from step 2. This extraction continues till all the leaf nodes are obtained.

The working of the interface to encode the data in the DNA stand is presented in Sections 3.1, 3.2, 3.3, 3.4.

4.1. XML File Encryption in a DNA Strand

That content of XML is encoded in a way that will avoid malicious attacks. As stated in the

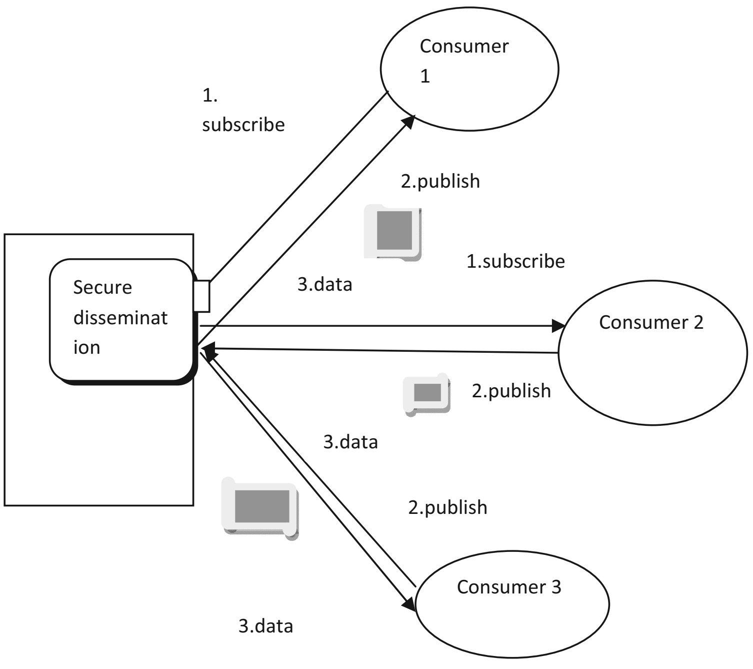


Figure 3. Architecture diagram of the secure dissemination interface.

introduction, DNA encryption has been chosen for encoding the data. The DNA molecule is composed of four basic groups of A, C, G and T.

The encoding problem can be described as follows:

$\sum\{A, T, C, G\}$ is a set defined as Z, the length is n. $Z = \sum^n = \{< b_1, b_2, b_3, \ldots, b_n > |bi € \sum i = 1, 2, \ldots, n\}(|Z| = 4^n)$ [11].

Thus each XML node's content is treated as a text; each character of the text is parsed and encoded using computer-mapped character encoding like UTF, ASCII.

Illustration: Encoding of a character in the text in the form of DNA strand

Here, we illustrate encoding for the character 'A'. The ASCII value is considered and its equivalent binary number equivalent is generated.

The procedure is outlined below:

ASCII value is considered and its equivalent binary number is generated. Let's take an example

A $\Longrightarrow$ its ASCII value is 65

We now convert the ASCII value into its equivalent binary number 100001. This binary number is encoded in the form of DNA strand.

Among the eight kinds of patterns, we have chosen a representation as shown in Table 1 to explain the method of encryption:

Binary number	DNA encoding
00	A
01	C
10	T
11	G

Table 1. Representation to map DNA nitrogenous bases to binary number.

Using Table 1, the encoded DNA strand looks like the following:

$$\begin{tikzcd}
10 \arrow[d] & 00 \arrow[d] & 00 \arrow[d] & 01 \arrow[d] \\
T & A & A & G
\end{tikzcd}$$

Thus the nucleotide TAAG is the encrypted form of A.

Following the above explained procedure of encryption the word "document" is encoded:

01000100 01001111 01000011 01010101
01001101 01000101 01001110 01010100

CACACAGGCAAGCCCCCAGCCACCCAGTCCCA

Thus after encryption every data of node will look like: $\sum$ {A, T, C, G}.

To add an additional level of encryption, we recommend XORing the data with a key. Each character can be XORed with an 8 bit key, which will be known to all the consumers.

An out-house intruder has to try 2^8 apart from knowing SRE and ERE to decode a data.

The same word "document" if XORed by a key 10011011 as shown above becomes The DNA strand after encryption now appears to be

GCGGGCCA GCTAGATAGCCAGCGT GCAC GAGG

4.2. Assigning a Starting and Restriction Enzyme to each Consumer

As referred in Section 2, the property of restriction enzyme to identify the recognition site is used to specify the access policy. Thus two restriction enzymes are randomly assigned from the database of 4000 restriction enzymes to each consumer subscribed for the XML data. This restriction enzyme is prefixed and suffixed to data as an annotation specifying which consumer is authorized to access the data.One which is prefixed is called SRE and the other one is called ERE.

Original form	01000100 01001111 01000011 01010101 01001101 01000101 01001110 01010100
Key	10011011 10011011 10011011 10011011 10011011 10011011 10011011 10011011
XOR	11011111 11010100 11011000 11001000 11010100 11011110 11010001 11001111

Name of the consumer	SRE		ERE	
	Name	Recognition site	Name	Recognition site
x	EcoRI-	GAATTC	EcoRII	CCWGG
y	SmaI	CCCGGG	Sau3A	GATC
z	EcoRV	GATATC	KpnI	GGTACC

Table 2. SRE and ERE assigned to the various consumers.

Considering Figure 1, there are three consumers subscribed to the document. x, y, z are the names assigned to them and their respective SRE and ERE are shown in Table 2.

SRE and ERE of consumers x and y are mathematically represented as $\langle$ SRE$_x$, ERE$_x\rangle$ and $\langle$ SRE$_y$, ERE$_y\rangle$ respectively. When a data is hidden in a DNA strand, it's actually annoted with the consumer's SRE and ERE to signify that consumer x has the authorization to access the node and the subtree. If consumer x is authorized to access the root node, the strand appears to be

GAATTC GCGGGCCA GCTAGATAGCCAGCGT GCAC GAGG CCWGG

If there are more than one subscriber of a node, for example x and y have subscribed for the root node, then any one pair of the SRE and ERE is used for annotation and the other consumer is informed about the new pair of SRE and ERE.

4.3. Scattering of Data in the Garbage File

A garbage file F can be defined as a large text file consisting of random combinations of A, C, T, G. DNA cryptography has an advantage that it provides four options for each bit as it in Quaternary. Thus the numbers of combinations are 4 for each bit. Thus for n bits the number of comparisons can be shown as

The total number of combination is as follows

$$C(1) = 4;\ \text{the enzyme with length 1 only}$$
$$C(2) = \binom{4}{1} * \binom{4}{1} = 4 * 4 = 16$$
$$\vdots$$
$$C(n) = \binom{4}{1} * \binom{4}{1} * \binom{4}{1} * \ldots * \binom{4}{1} = 4^n$$

Now the terms

$$4+4^2+4^3+4^2+4^5+\ldots+4^n = \frac{4(4^{n+1}-1)}{3}$$

Thus if n is 128 bits, the number of combinations is 9.88E+78.

And in comparison to binary the number of combinations would have been

$$C(1) = 2;\ \text{the enzyme with length 1 only}$$
$$C(2) = \binom{2}{1} * \binom{2}{1} = 2 * 2 = 4$$
$$C(3) = \binom{2}{1} * \binom{2}{1} * \binom{2}{1} = 8$$
$$\vdots$$
$$C(n) = \binom{2}{1} * \binom{2}{1} * \binom{2}{1} * \ldots * \binom{2}{1} = 2^n$$

Now the terms

$$2+2^2+2^3+2^2+2^5+\ldots+2^n = 2(2^n-1)$$

The number of comparisons would have been 6.806E+38

Figure 4 shows a graphical representation of the number of bits versus the number of combinations in terms of exponentiation. It can be seen that the values of binary is lower than quaternary. Table 3 shows number of combinations versus number of bits for both quaternary and binary number system.

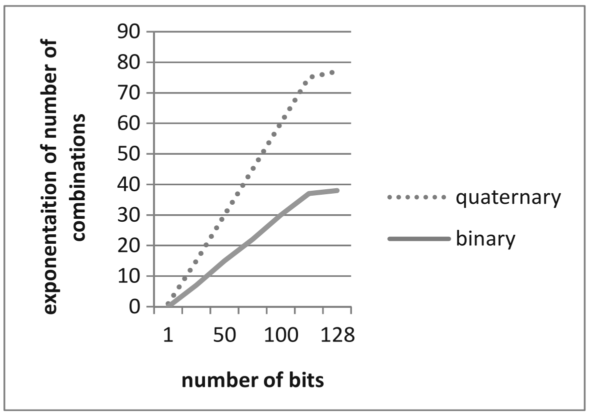


Figure 4. Number of bits versus number of combinations in terms of exponentiation.

The data is embedded into the garbage file at random locations. A table *Tab* is also maintained to store the information of the address of the node in the file. The location is calculated as number of words from starting of the file (length of each word is 3 characters). The

No of bits	1	25	50	75	100	125	128
quaternary	20	6E+15	6.76E+30	7.61E+45	8.57E+60	9.65E+75	6.18E+77
binary	2	67108862	2.25E+15	7.56E+22	2.54E+30	8.51E+37	6.8E+38

Table 3. Number of combination versus number combinations in quaternary and binary number systems.

attributes of table *Tab* are ⟨ name of the node N, address addr of the node, length L of the data in the node⟩ when encrypted using the method mentioned in Section 3.1.

The address addr_N is assigned randomly such that

$$\text{addresss addr}_N = \{\text{addr}|\text{addr} < \text{addr}_i \quad \text{addr} > \text{addr}_i + l_i\}$$

where $1 \leq i \leq n - 1$.

l_i is the string length of the data at node i.

The node is stored in the file and the address of the node is appended beside data of the parent, so that, while traversing, if one knows the node, he can find all the children of the node in the garbage file. The location is also encrypted in the format as mentioned in Section 3.1.

SRE	Data in the encrypted form	Address of leftmost child	...	Address of rightmost child	ERE

Figure 5. Diagrammatic representation of the node.

Now the strands will have a representation as shown in Figure 5, stating that the whole subtree T starting from the node is visible to the client X.

GAATTC GCGGGCCA GCTAGATAGCCAGCGT GCAC GAGG ACTGGGTACCATG CCWGG

SRE DATA of the node ADDRESS of the child ERE

If T2 is the subset of T_1 and it has to be sent to consumer y, then head of sub tree T_2 will be prefixed and suffixed with ⟨ SRE_y, ERE_y⟩ , thus signifying that consumer y will only be able to locate the head of T_2 and its children, but will not be able to search the address of its parent node in the file. The proposed secure dissemination technique is based on publish/subscribe. Whenever a consumer subscribes for the data, the required file F' is sent to him.

The consumer will first locate the data starting with SRE and ERE. Then he will try to extract the data and child's addresses and unfold the rest of the tree till a leaf is encountered.

5. Proposed Algorithm of the Secure Dissemination

The working of the algorithm for secure dissemination which implements the technique explained in Section 4 involves the following two tasks:

- Execution at the server end as producer of the file
- Execution at the consumer end to construct the sub tree

The requirements of this approach are that the consumers should subscribe to the interface, and the administrator at the server end should mention the access policies of the consumer. Depending on the trust it has, it will then customize at the server end, which user can access which part of data. The XML file to be transmitted is then prepared according to the algorithm and sent over the network.

5.1. The Algorithm Followed at Server End who is the Producer of the File

```
1. Initialization:
   • Generate a garbage file F = $\sum\{A,T,C,G\}$ is prepared
2. XML FILE is parsed and each tag is traversed using preorder traversal
   • r=preorder(T)
   For each node r
      an entry is maintained in the table T
      $A_N$ =randomaddress( )
   /* randomly assign an address to each node to be inserted in the garbage file F in the garbage file, so that it doesn't match with any earlier assigned address or any earlier assigned address + length of the data.
   Add (⟨T, Name of the node r, Address assigned to the node $A_N$, Length of the data of the traversed node⟩ */
3. D=r $\Longrightarrow$ info //D is the info at node
4. $E_x$ ($D_N$) =encrypt(D) // Perform DNA encryption as explained in the paper
5. List=ADD($E_x$($D_N$)// maintain a list of data elements and its encrypted value
6. GetSubscribedConsumer( )//For each client
   • LIST 1=add(SRE,ERE,n) in the table Enzymes // SRE and ERE to each consumer $x$ so that the recognition site doesn't exit in the data.
     $n$ number of bits of SRE and ERE.
7. X'=prepXMLTemp() //Parse the XML file and according to access policy prefixing and postfixing for the XML tags which the consumer is authorized.
8. F'=Modify(X') //The server then parses temporary XML file X' checks out if an tag is prefixed with for then SRE is picked from the table LIST1 and prefixes to $E_x$ ($D_N$). The data now transforms into $SRE_x$ E ($D_N$), where X is the consumer number. Then the address location of the children is suffixed after looking into the table T. The data now becomes

   $SRE_x$ E ($D_N$) $A_{\text{leftmostchild of X}}, \ldots, A_{\text{rightmostchild of X}}$.

   At the end the data is suffixed with $ERE_x$ to show the reading of the data to end here. The data now becomes $SRE_x$ E ($D_N$) $A_{\text{leftmostchild of X}}, \ldots, A_{\text{rightmostchild of X}}$ $ERE_x$. The data is now embedded into the location which was entered in table T.
9. Transmit the prepared file F' to all the consumers
```

5.2. The Algorithm Followed at Consumer End to Construct the Subtree

```
1. loc=Search(F', SRE) //Scan the SRE allocated to the consumer by the Server in the received garbage file F'. The position of last bit of SRE in F' is called loc
2. loc1=Search(F', SRE) //Scan the ERE allocated to the consumer by the Server in the received garbage file F'. The position of last bit of ERE is called loc1
3. string1= readcontent(loc, loc1,F') //Read all the characters between loc and loc1 in string1. String1 contains

      E($D_N$) $A_{\text{leftmostchild of N}}, \ldots, A_{\text{rightmostchild of N}}$

4. E ($D_N$) =Extractdata (string1) //Extract encrypted data from string1. Where N is the root of the subtree which the consumer is authorized to access.
5. Data= Decrypt(E ($D_N$)) //Decrypt the encoded data
6. T=Root(data) //insert data as root node in subtree
7. Repeat till all leafnodes are reached
     A=Extractaddresses(string1)
   //For each address ($A_{\text{leftmostchild of X}}, \ldots, A_{\text{ rightmostchild of X}}$)
     string1=readcontent(loc,loc1,F')
     E ($D_X$ =Extractdata (string1)
   //Where X is the node extracted
       Data=Decrypt(E ($D_X$))
       Insert(T,data,X)
   //insert data as a child under the node X
8. Return (T)
```

6. Interface for Dissemination

The multicast dissemination interface is built dynamically and asynchronously using a publish-subscribe methodology. Once the consumer subscribes with the interface, it will be assigned a SRE and ERE.

A temporary XML document is prepared by prefixing and postfixing the XML tags which the consumer is authorized as shown in Figure 6. If a consumer is allowed to visit a top level tag, then he/she is eligible to see all the child tags. If there are more than one subscriber of a node, for example x and y have subscribed for the root node, then any one pair of the SRE and ERE is used for annotation and the other consumer is informed about the new pair of SRE and ERE.

The interface starts processing this temporary XML document and later provides the following two options:

• Produces one single file to send to all clients.

In this case the interface chooses as many SRE and ERE as there are "for" attribute in the temporary XML file. Now before the start tag of each tag having "for" attribute the software will append both information.

```
<document>
  <last_updated  for="consumer3">July 28, 2012
    </last_updated>
  <copyright for="consumer2">ABC company
    </copyright>
  <maintainer email="ras@juitcom"
    url="http://www.juit.com/">
  <name for="consumer1">
    <first_name>marko</first_name>
    <middle_name>D</middle_name>
    <last_name>Harold</last_name>
  </name>
  </maintainer>
  </document>
```

Figure 6. Temporary XML file prepared.

• Produces separate file for each set of clients

The interfaces will produce as many encoded files as there are “for” attributes in the temporary file. In this example there will be three output files. Each encoding is done with different set of SRE and ERE. The interface will give the name of SRE and ERE used for that particular file.

7. Security Analysis of Proposed Secure Dissemination Technique

The security analysis of secure dissemination technique and the algorithm are discussed based on three points of views

- Probability of getting the right SRE and ERE
- Time taken to find the right SRE and ERE
- Requirement satisfaction

7.1. Probability of Getting the Right SRE and ERE

If we consider the classical DNA encryption then there is a dataset of 4000 restriction enzyme. The number of options is $\binom{4000}{1} = 4000$. We can have at most $4000 * 4000$ number of combinations for choosing both SRE and ERE from a dataset of restriction enzymes. Hence the probability of getting the right set of enzymes is

$$p(A) = \frac{1}{(4000)^2}$$

This limits the power of DNA encryption as it is easy to find the right set of restriction enzymes. Thus the probability of finding the restriction enzymes is high. Therefore, it is important to increase the number of combinations. Hence we are not constraining the choice of restriction enzymes to classical dataset of restriction enzymes. We are customizing an enzyme in the following fashion to lower the probability to guess the right set of enzymes. The enzyme can be any combination of A, C, T, G which can span up to any length.

In the light of the above definition, let us compute the probability of guessing the right set of enzyme:

The total number of combinations is as follows:

$$4 + 4^2 + 4^3 + 4^2 + 4^5 + \ldots + 4^n$$

So if the length enzyme is taken up to n length then the probability is

$$P(A) = \frac{1}{(4 + 4^2 + 4^3 + 4^2 + 4^5 + \ldots + 4^n)}$$

Hence the probability becomes

$$p(A) = \frac{3}{4(4^{n+1} - 1)}.$$

Taking $n = 5$ we get $p(A) = \frac{3}{4(4^6 - 1)} = 0.0001831$.

Figure 7 shows the probability of getting the right combination with respect to number of bits. It can be seen that the expression is highly convergent to zero.

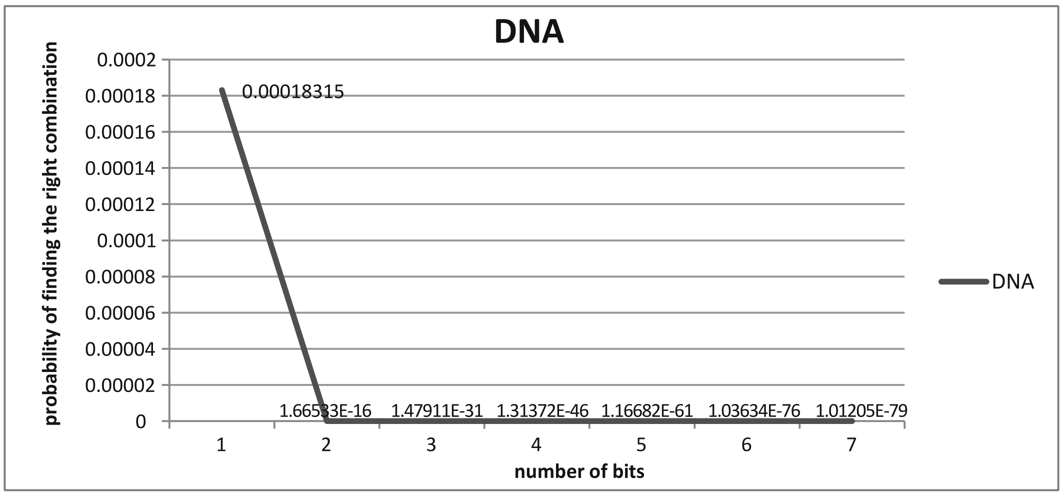


Figure 7. The number of bits versus the probability to find the right combination.

7.2. Time Taken to Find the Right SRE and ERE

Time taken to find the right SRE or ERE depends on checking all possible key combinations until the correct key is found. This is also called Brute-force attack where systematically all possible key combinations are checked [20]. The calculation of the time taken is as follows:

Let's assume Faster supercomputer: 10.51 Pentaflops= 10.51E+15Flops (Floating point operations per second)

Assume that Number of Flops required per combination check is 1000

No. of combination checks per second= (10.51E+15)/1000 = 10.51E+12

No. of seconds in one year= $365 \times 24 \times 60 \times 60 = 31536000$. If $\propto$ is the number of combinations of a technique then number of the years to crack $= \dfrac{\propto}{(10.51\text{E} + 12) \times 31536000}$.

Substituting the value of $\propto$ for different symmetric cryptosystem is shown in Table 4.

It can be observed from the Table 4 and Figure 8 that the time taken to crack the right key/restriction enzyme is high in comparison to other symmetric cryptosystems. Other symmetric approaches will have scalability issues unless key distribution and key strength are systematically enforced.

But the proposed algorithm will have no scalability issues as the number of files created for various consumers is only one and the key id also assigned once when the consumer subscribes to the interface.

The length of the key is also variable, making it more difficult to decrypt as the number of combination is equal to 9.881E+78, which shows the robustness of the proposed technique.

Key size	DES (128 bits)	DES (168 bits)	AES (128 bits)	AES (192 bits)	AES (256 bits)	DNA 128bits **(Proposed)**
Possible combinations ($\propto$)	7.2E+16	3.7E+50	3.403E+38	6.2E+57	1.1E+77	9.881E+78
Time to crack (years)	6850secs	1.11E+30	1.02E+18	1.87E+37	3.31E+37	2.98E+58

Table 4. Time required cracking the combination in various types of symmetric cryptosystem.

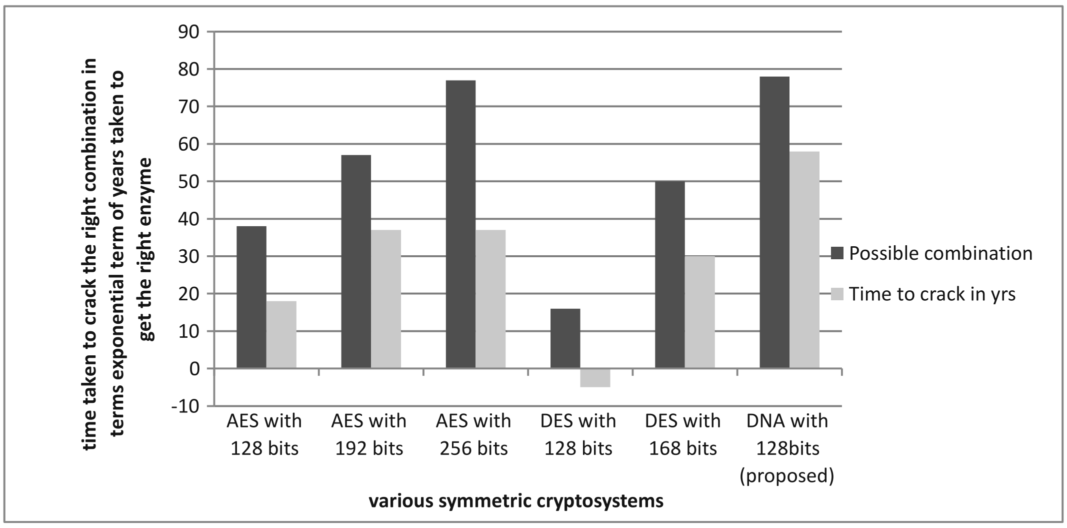


Figure 8. Possible combination and year to crack the right combination with respect to the various symmetric cryptosystem.

7.3. Requirement Satisfaction

The requirement of the problem was to ensure integrity, confidentiality and access control. Integrity is obtained as the probability of cracking the data is low, as shown in the above section. Confidentiality and access control are also obtained as the consumer will be able to access data which he has subscribed to by the interface.

The proposed technique support underlying n-ary tree structure of the XML document and hides the data in the form of linked list in a garbage file. The overhead is less as a single file is sent to all the consumers. The disadvantage of the solutions given by [2, 4] is that if the Local XML structure changes, it requires associated routing topology to be changed. Thus a subscriber needs to have a prior knowledge of the routing structure as the router cannot fetch any content which is not hosted currently. We have implemented at server level, thus any change is just mentioned at the interface level, reducing the cost and time required for a change in the system. This approach will not have scalability issues as other symmetric crypto systems. In [21, 27] the requestor asks for a set of concepts and therefore requires knowledge of the ontological structure while in our work the ontological part is transparent for the requestor as the focus is on general access control.

Finally, our work is related to the secure XML broadcasting problem where the focus is the secure dissemination of XML documents to authorized users. In our approach we will be specifying access policies using variable length key inspired from real world restriction enzymes, which is a computationally secure technique.

8. Conclusion

Finally, to summarize, secure dissemination technique presented in this paper ensures that the consumers of the data are the legitimate ones according to the access policies. It presents a computationally secure technique in which there is a possibility to break the system theoretically, but it's infeasible to do so by any known practical means. A multicast dissemination interface at the server/producer end is proposed to implement the secure dissemination technique. Each client/consumer will subscribe in the interface and then automatically will be assigned a pair of randomly generated restriction enzymes called as SRE and ERE. The data will be appended with the SRE and ERE to signify that the data is meant for the respective consumer who has been assigned a particular SRE and ERE. Later the data is encrypted according to the technique and scattered in the garbage file. The garbage file is then transported to all the consumers where they will be able to view only the data as per the access control policies.

In particular, this paper explains how to secure the data in a DNA strand and provide data integrity. It has been also proved that due to the quaternary number system followed in the technique, it has very low probability of cracking the key. The number of years to crack the combination is also very high in comparison to various pre-existing symmetric cryptosystems like DES and AES. The results indicate that the proposed technique not only satisfies the requirement specification of secure dissemination, but also points out its robustness in terms of time required to break the key. The time to crack the key is quite long and increases with the increase in key length thus proving it to be computationally hard to crack by any known practical means.

We plan to further investigate and find out how other access control policies and integrity models can be implemented using the proposed secure dissemination technique.

References

[1] N. Gruschka, M. Jensen, L. Lo Iacono, N. Luttenberger, Server-Side Streaming Processing of WS-security. *IEEE Transactions on Services Computing* Issue 99 (06 January 2011), pp. 1–14. Digital Object Identifier: 10.1109/TSC.2010.61

[2] E. Bertino, E. Ferrari, Secure and selective dissemination of XML documents. *ACM Transactions on Information and System Security - TISSEC*, **5**(3) (2002), pp. 290–331.

[3] `http://www.w3schools.com/dtd/dtd_intro.asp` as on july 2012

[4] A. Kundu, E. Bertino, A New Model for Secure Dissemination of XML Content. *IEEE Transactions on Systems, Man, and Cybernetics – part C: Applications and Reviews*, **38**(3) (May 2008).

[5] L. O'BRIEN, P. MERSON, L. BASS, Quality Attributes for Service-Oriented Architectures. *International Workshop on Systems Development in SOA Environments (SDSOA'07: ICSE Workshops 2007)*, 2007, pp. 3.

[6] D. E. KNUTH, *Art of Computer Programming, Volume 1: Fundamental Algorithms.* (3rd Edition)

[7] `http://dosattack.net/` as on july 2012

[8] A. ADHIKARI, DNA Secret Sharing. *IEEE Congress on Evolutionary Computation*, (July 16-21, 2006) Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, pp. 1407–1411.

[9] B. ROY, G. RAKSHIT, P. SINGHA, A. MAJUMDER, D. DATTA, An improved Symmetric key cryptography with DNA based strong cipher. *International Conference on Devices and Communications – ICDeCom*, (2011) pp. 1–5.

[10] A. GEHANI, T. LABEAN, J. REIF, DNA-based cryptography. Lecture Notes in Computer Science, vol. 2950, pp. 167–188, 2002.

[11] S. V. KARTALOPOULOS, DNA-inspired cryptographic method in optical communications, authentication and data mimicking. *Proc. of the IEEE on Military Communications Conference*, (2005) vol. 2, pp. 772–779.

[12] K. TANAKA, A. OKAMOTO, I. SAITO, Public-key system using DNA as a one-way function for key distribution. *Biosystems*, **81**(1) (2005), pp. 25–29.

[13] M. SAEB, E. EL-ABD, M. E. EL-ZANATY, On Covert Data Communication Channels Employing DNA Recombinant and Mutagenesis-based Steganographic Techniques. *International Conference on Computer Engineering and Applications*, World Scientific and Engineering Academy and Society (WSEAS) Stevens Point, Wisconsin, USA, pp. 200–206.

[14] G. CUI, L. QIN, Y. WANG, X. ZHANG, Information Security Technology Based on DNA Computing. *International Workshop on Anti-counterfeiting, Security, Identification, IEEE*, (2007) Xiamen, Fujian, pp. 288–291.

[15] X. WANG, Q. ZHANG, DNA computing-based cryptography. *Fourth International Conference on Bio-Inspired Computing*, (2009) BIC-TA '09. Publication Year: 2009, pp. 1–3.

[16] `http://rebase.neb.com/rebase/rebase.enz.html` as on august 2012

[17] `http://www.neb.com/nebecomm/products/category1.asp` as on august 2012

[18] R. C. CHALMERS, K. C. ALMEROTH, On the Topology of Multicast Trees. *IEEE/ACM Transactions on Networking*, **11**(1) (2003), pp. 153–165.

[19] G. CUI, L. QIN, Y. WANG, X. ZHANG, An Encryption Scheme Using DNA Technology. *3rd International Conference on Bio-Inspired Computing: Theories and Applications*, (2008), BICTA 2008. Publication Year: 2008, pp. 37–42.

[20] `http://www.eetimes.com/design/embedded-internet-design/4372428/How-secure-is-AES-against-brute-force-attacks-` as on august 2012

[21] M. A. RAHAMAN, Y. ROUDIER, P. MISELDINE, A. SCHAAD, Ontology-based Secure XML Content Distribution. *Proceedings of the 24th International Information Security Conference*, (May 2009) Pafos, Cyprus, pp. 294–306.

[22] W.-C. L. BO LUO, D. LEE, P. LIU, *A Flexible Framework for Architecting XML Access Control Enforcement Mechanisms.* Vol. 3178/2004 of Lecture Notes in Computer Science. Springer Berlin/Heidelberg, December 2004.

[23] E. DAMIANI, S. D. C. DI VIMERCATI, S. PARABOSCHI, P. SAMARATI, Fine Grained Access Control for SOAP Eservices. *In Proceedings of the 10th International Conference on World Wide Web*, (2001) New York, NY, USA, pp. 504–513.

[24] W. FAN, C.-Y. CHAN, M. GAROFALAKIS, Secure XML Querying With Security Views. *In Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, (2004) New York, USA, pp. 587-598.

[25] G. KUPER, F. MASSACCI, N. RASSADKO, Generalized XML Security Views. *In Proceedings of the Tenth ACM symposium on Access Control Models and Technologies*, (2005) New York, NY, USA, pp. 77–84.

[26] M. MURATA, A. TOZAWA, M. KUDO, S. HADA, XML Access Control Using Static Analysis. *Proceedings of the 10th ACM conference on Computer and Communications Security*, (2003) New York, USA, pp. 73–84.

[27] A. ROTA, S. SHORT, M. A. RAHAMAN, XML secure views using semantic access control. *Proceedings of the 2010 EDBT/ICDT Workshops.*

Received: January, 2013
Revised: August, 2013
Accepted: September, 2013

Contact addresses:

Rajni Mohana
Department of CSE and ICT
Jaypee University of Information Technology
Wakhnaghat
India
e-mail: `rajnivimalpaul@gmail.com`

Deepak Dahiya
Department of CSE and ICT
Jaypee University of Information Technology
Wakhnaghat
India
e-mail: `deepak.dahiya@juit.ac.in`

RAJNI MOHANA is working as Assistant Professor in the Department of CSE and ICT at Jaypee University of Information Technology (JUIT), Waknaghat, India. She is persuing her Ph.D from Jaypee University of Information Technology (JUIT), Waknaghat, India under the guidance of Dr. Dahiya. Her area of interest is service oriented architecture, cloud computing and she has over 10yrs of experience in teaching.

DEEPAK DAHIYA is currently working as Professor in the Department of CSE and ICT at Jaypee University of Information Technology (JUIT), Waknaghat, India. Deepak has over 20 years of extensive experience in IT Industry and Academics in India, Australia, US, UK and Oman. He is also a reviewer for various renowned journals from Elsevier, IET and Wiley. Deepak is a Visiting Researcher to RMIT University, Australia, Guest Faculty to FMS Delhi, IIM Rohtak, IIM Kozhikode and LNMIIT Jaipur. He has conducted senior executive training programmes for both private and government sectors. In the IT Industry, he has been consulted by corporate clients in UK, US and India. He is a senior member of IEEE and life member of the Computer Society of India.