Journal of Theoretical and Applied Information Technology
31st August 2022. Vol.100. No 16
© 2022 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SECURE XML VIA APPLICATION OF SOAP-BASED TEXTUAL STEGANOGRAPHY ALGORITHM

[1]MUBARAK AL-MARRI, [2]LOKMAN MOHD FADZIL

[1]PhD Student, National Advanced IPv6 Center (NAv6), Universiti Sains Malaysia, Penang, Malaysia

[2]Senior Lecturer, National Advanced IPv6 Center (NAv6), Universiti Sains Malaysia, Penang, Malaysia

E-mail: [1]mubarakalmerri@student.usm.my, [2]lokman.mohd.fadzil@usm.my

ABSTRACT

This paper briefly reviews the emerging use of and the application of steganographic techniques for exchanging secret messages in a digital medium. In the era of ever-increasing cyberattacks, there is an extreme need for reliable and secure communication. In this respect, cryptography approach has been used to encrypt or encode a textual message with a protective key to prevent unauthorized access. However, encrypted information can conspicuously challenge the hackers and instigate them to crack the code. In contrast, stenographic methods are surreptitiously employed on what appears as a perfectly normal text is actually hiding certain secretive message contents. In a two-part communication structure comprising the message and the message transport, the message is hidden by applying steganographic techniques to be delivered by Simple Object Access Protocol (SOAP). The author’s contribution is a proposed algorithm that resolved current algorithm limitations by increasing message size, maintaining the stego file size equivalency to the real file, and minimizing the real message and the stego message differences. This is accomplished by developing encoding and decoding functions in the algorithm. The encoder extracts the equivalent sequence number from a predefined file representing a null value to generate an arbitrary record to be written to an XML file. The decoder read and select the block to extract the equivalent number that equals the null tags, and search in the database to select the equivalent character to the extracted number to be added to the string. Optional SOAP fields which are normally empty or unfilled are also being used with other tags to hide one character, so one block will be equal to one character based on null values. The XML file, embedded with the SOAP tags, which provides competitive performance in benchmark tests, can be used to securely transact information over the Web.

Keywords: *Algorithm, Cryptography, SOAP, Steganography, XML*

1. INTRODUCTION

The communication and data sharing between applications, called inter-process communication (IPC), enables applications to use them as clients that requests service from a number of other application or process, or servers that responds to a client request.

Depending on the usage, an IPC can be designed as a Graphical User Interface (GUI) or a console application to communicate with other applications running on the Windows- or UNIX- or other operating systems-based local or networked computers.

Clipboard, Component Object Model (COM), Data Copy, Dynamic Data Exchange (DDE), File Mapping, Mailslots, Pipes and Remote Procedure Call (RPC) and Windows Sockets IPC are data exchange mechanisms currently supported by Windows [1].

There are numerous applications developed for exchanging data using Web services. One example are banks communicating with their customers for exchanging financial instruments. Others include companies synchronizing with their customers and suppliers for demand orders, packing lists, air waybills, insurance documents, invoices, among others.

To this end, many solutions are available to manage these types of transactions, but they are not without limitations. One example would be an IPC making use of software interface called Distributed Component Object Model (DCOM) or Common Object Request Broker Architecture (CORBA), the middleware architectures with programming language portability and platform independence in managing Internet communications [2].

Nevertheless, these protocols communications sometimes fail as a properly-configured firewall opens a very few ad-hoc ports to function securely. The same case goes for HTTP server, which always listens to port 80 [3].

However, DCOM necessitates initial communication handshake via port 135, in addition to opening a range of ports to run DCOM objects hosting processes as well as MTS/COM+ server applications [3].

Similarly, a CORBA server object does not necessarily function via a specially designated port, because CORBA applications and its related Object Request Brokers (ORBs) generally use objects launched at arbitrarily selected ports [4].

Standardized firewalls with static configurations are not suitable for dynamic DCOM and CORBA applications because their functionality is based on the notion that a client-server pair will always use a fixed, designated port [4].

Based on this fact, Simple Object Access Protocol (SOAP) was developed as an industry standard Extensible Markup Language or XML-based Internet-friendly protocol using Hypertext Transfer Protocol (HTTP) as its preferred protocol using port 80 address.

Since an XML parser, which functions to search and process XML documents, is practically available on any operating system, SOAP will be the future Internet-firewall-friendly, interoperable protocol web services access protocol [3].

In this paper, data exchange security issues are being investigated with a proposal for a steganographic algorithm to be used in an XML-SOAP mechanism.

Steganographic methods three foundational characteristics are capacity, imperceptibility and security, which are necessary to disguise classified information [5]. Based on an extensive investigation paper which surveys 50 papers on steganographic techniques, this paper's problem statement focuses on maintaining minimum stego file size as well as establishing trusted transmission channels to minimize the risk of cyber attack [6].

To this end, the research objective is to propose and develop an algorithm to minimize the use of text characters using Binomial Coefficient Theory to encode characters into code for disguise for transmission via SOAP-XML transport mechanisms with eventual decoding.

This paper is structured as the following: the subsequent section attempts to educate the readers with available research on SOAP methodology, with focus on the use of steganography techniques in literature. In section three, a steganographic algorithm is proposed and explained for use with SOAP. Section four focuses on the experiments and discussion of the results. The fifth section summarizes the conclusions and perspectives derived from this paper.

2. LITERATURE REVIEW

2.1 SOAP Overview

SOAP, a universally-supported HTTP-based protocol for simple, powerful and extensible communication between applications achieved W3C Recommendation status on June 24, 2003 [7].

As an independent protocol, SOAP establishes anywhere, anyplace communication between any platform, language, or server as the precedent layer for increasingly intricate web services, such as Universal Description Discovery and Integration (UDDI), XML schema and Web Services Description Language (WSDL) [8].

A SOAP message is a structured XML document comprising an envelope element that identifies the XML document as a SOAP message. It also contains a header element that contains header information, a body element that contains call and response information, and a fault element containing errors and status information [9]. A skeleton SOAP message is illustrated in Figure 1.

```
<?xml version="1.0"?>
<soap:Envelope
     xmlns:soap="http://www.w3.org/2001/12/soap-
     envelope"
     soap:encodingStyle="http://www.w3.org/2001/1
     2/soap-encoding">
     <soap:Header>
          ...
     </soap:Header>
     <soap:Body>
          ...
     <soap:Fault>
               ...
     </soap:Fault>
     </soap:Body>
</soap:Envelope
```

Figure 1: SOAP XML-Based Message Structure [7]

The SOAP envelope element is shown in Figure 2.

```
<soap:Envelope
                  xmlns:soap="http://www.w3.org/2001/
                  12/soap-envelope"
                  soap:encodingStyle="http://www.w3.o
                  rg/2001/12/soap-encoding">
                       ...
                  >>Message information goes here<<
                ...
                  </soap:Envelope>
<soap:Envelope
                  xmlns:soap="http://www.w3.org/2001/
                  12/soap-envelope"
                  soap:encodingStyle="http://www.w3.o
                  rg/2001/12/soap-encoding">
                       ...
                  >>Message information goes here<<
                ...
                  </soap:Envelope>
```

Figure 2: SOAP Envelope Element [7]

The xmlns:soap namespace embodies the value of "http://www.w3.org/2001/12/soap-envelope". The data types used in the document are defined by the encodingStyle attribute which appear in an arbitrary SOAP element related to the content and child elements. A SOAP message has no default encoding [9].

Syntax

soap:encodingStyle="URI"

The optional SOAP header element contains application-specific information, such as authentication and payment, for example, about the SOAP message.

The presence of the Header element indicates the Envelope element's first child element. Note: All immediate child elements of the Header element must be namespace-qualified [9]. An example of SOAP header is illustrated in Figure 3.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-
envelope"
soap:encodingStyle="http://www.w3.org/2001/
12/soap-encoding">
<soap:Header>
<m: Title
xmlns:m=http://www.myserver.com/title/
soap:mustUnderstand="A">BCD
</m: Title >
</soap:Header>
...
</soap:Envelope>
```

Figure 3: SOAP Header Example

Per SOAP syntax rules, a SOAP message must be encoded using XML, use the SOAP envelope and encoding namespaces, and must not contain both the DTD (Document Type Declaration) reference and XML processing instructions [9].

2.2 Steganography

Steganography is the science of hiding messages through obscurity methods. Only the sender and the intended recipients know about the message's existence [10].

Security through obscurity is achieved by using secrecy of a design or an implementation. Such systems may have theoretical or actual security vulnerabilities, but if they are not known, then the likelihood is that these flaws will never be found.

Security through obscurity has never achieved formal acceptance for systems security implementation since it contradicts Keep It Simple principle (Kerckhoffs's principle).

Steganography, contrary to cryptography, protects both the message contents and the communicating parties. It lacks attractability, or the ability to attract attention, in contrast to the 'supposedly unbreakable notion' aura of encrypted messages that stir suspicion and challenge potential hackers. Simply, steganography hides the message so that it is unseen, while cryptography scrambles the message so that it cannot be understood [10].

The use of steganography trailed back to 440 BC when ancient Greeks first found the means to hide messages by shaving and tattooing the message onto the slaves' heads. The slaves would be sent to deliver the message when the slaves' grown hair hide the tattoo, with their heads shaved upon arrival to reveal the message [11].

Another creative steganographic method was to transcribe messages on wooden panels and covered with wax. Since wax tablets were commonly used as reusable writing surfaces during that era, secretive communication was the least expected when the wax on the tablets was actually melted to reveal the hidden message [12].

World War II saw ingenious methods including using invisible ink and microdots to hide messages, where the message was embedded within a body of a larger plain text using null cipher method [13].

A digital image file forms an array of binary numbers, each of them represents a pixel. To embed or hide a message inside an image file using the steganographic technique [14], two files are required.

The first file is the image file that will embed the message, whereas the second file is the message to be hidden or embedded, producing the so-called stego-image file. Common approaches used to hide a message in digital images are listed in the subsequent section.

2.3 Least Significant Bit Insertion

It is a common yet straightforward way to embed messages in digital image files. The technique is to modify the least significant bit of the pixel array. Unfortunately, this method is vulnerable to even small manipulations [15].

2.4 Masking and Filtering

This technique hides messages by marking the image in a way similar to the watermarking method. This is accomplished by having certain pixel values raised or lowered by a numeric percentage in masked areas [15].

2.5 Algorithms and Transformation

Both the steganographic method and the SOAP protocol provide a perfect answer to the secret message equation. Given the fact that there are a number of methods of hiding text, either using video, image, or audio file, the focus of this paper is only on hiding a message in a text in this paper.

As such, any one of the three methods available can be used: white spaces (open space), punctuations (syntactic) and in the text itself (semantic). Once hidden, they appear to represent the whole contents of the text [16].

Since SOAP is based on XML, the XML-tags-equivalent SOAP tags can be utilized to hide secret bits. However, this apparent approach of hiding secret messages has limitations as there are only a few bits where the message can be hidden. The framework for steganography to work with SOAP messages is illustrated in Figure 4.

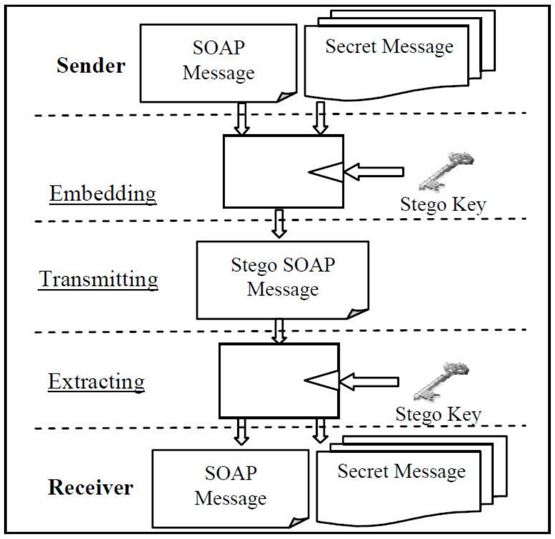


Figure 4: Steganographically Applied SOAP Message [16]

The so-called stego-SOAP message model consists of a SOAP message and a secret message, which functions initially via an embedding function, to acquire and combine a normal message and a secret message, resulting in a new SOAP-compliant message format.

This message contains the secret message based on the stego key works only on the sender. Next, the extracting function will use the key known only by the receiver to isolate the normal message from the secret message separately as the output. In situations that the provided key is incorrect, only the real message would be extracted.

2.6 Related Work In Literature

There is very minimal research on textual steganography compared with other related studies conducted on mainstream media such as images, videos, and audio.

A number of textual steganographic methods will be discussed in this section. Important to note that not all of these methods are applicable for structured documents, especially XML. This is due to the fact that structured documents' formatting rules require that any intended secret messages will need to follow the formatting rigorously.

A dynamic new method was proposed to utilize in-between words and paragraphs whitespaces in a right-justified text. Nevertheless, this method suffers from being weak and easily broken, apart from the cover text needs to be dynamically generated according to the secret message [17].

Shirali-Shahreza's method utilizes hiding texts based on UK and US English differences. For example, the US spelling "Program" is used to hide the value '0' while UK spelling "Programme" is used to hide the value '1' in his first method. In his second method, words with different terms but same meaning are used to hide messages.

For example, the word "Gas" in US English will be used to hide 0, while the equivalent word "Petrol" in UK English will be used to hide the value '1'. Again, these methods applicability is limited to only one bit of hidden data per substitution, which is easily detectable by spell-checkers [18, 19].

Liu et al.'s method hides online chat-based texts. Secret text is encoded by the transposition of interior adjacent letters of words, based on research that an arbitrary word's middle letters randomization has little or no effect on the ability of skilled readers to understand the text.

The application of online chat for steganography is based on the fact that typing mistakes is viewed as common due to speed typing, and the universal nature of chat sessions [20].

Inoue et al. proposed to embed secret data into XML documents to switch [21]:

- Between the representations of an empty element, one bit of data can be hidden per empty tag
- Between adding and removing white spaces before the tag's closing-bracket, one bit of data can be hidden per tag
- The elements' order where in exchange of two elements, one bit of data can be hidden
- The order of attributes within an element, one bit of data can be hidden per an exchange of two attributes
- By exchanging inner-tags with outer-tags that contain them, one bit of data can be hidden per an exchange [21]

Memon et al. proposed several steganographic techniques for the purpose of securing the cover file rather than hiding information, summarized as follows [22]:

- Inserting random characters inside all tags and values .and random characters are inserted after each nth character

- Shuffling XML tags based on position. The first tag is swapped with the n^{th} tag and the second tag with the $(n-1)^{th}$, and so forth

- Attribute-specific shuffling. Tags are shuffled based on the first character of attribute value of the root element

- Reverse characters. The sequence of characters in all tags is reversed [20]

Zhang et al. proposed a method to hide messages in SOAP files by employing SOAP file keywords as cover message to hide secret message bits. Each SOAP keywords character's case is mapped to the relevant bit in the secret message binary code, with one bit hidden per character [23].

Finally, Alrouh et al. proposed an improved method [24] based on method [19] in [21] and method [18] in [20] to hide messages in SOAP files. Steganography is accomplished by rearranging the specific XML elements' order according to the secret message.

This method's advantages are that the file size is intact, and the capacity is proportional to the number of elements. However, the fallback is that the generated stego-SOAP file is arguably suspicious, and applicable only when the elements' order is not important [16].

3. PROPOSED WORK

3.1 The Proposed Algorithm

A number of algorithms to hide a message in a text were reviewed with the most promising algorithm being the Alrouh's method [24]. The algorithm bypasses many limitations with additional features like increasing the message size, keeping the original stego file size, and decreasing the real message and the stego message differences.

For the Alrouh's method, one major issue has been identified. The permutation among tags in any "XML" file is not applicable, therefore, easily detected either by the reader directly or by a program. In addition, this algorithm does not produce the stego file appearing like the real file, which is one of steganography basic concepts. Alrouh also did not include the experimentation and implementation part in his work to demonstrate his work and the validity of his algorithm.

The proposed algorithm attempted to avoid these issues, limitations and problems seen in previous sections, and to maintain XML file primary features, such as the stego file visually equivalent to the real file. In reality, most data will not require all the information. Optional fields which are normally empty or unfilled will be used with other tags to hide one character, so one block equals to one character based on null values. Figure 5 demonstrates the algorithm with examples null Email and Mobile tags, which is equivalent to character "h".

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
xmlns="http://schemas.xmlsoap.org/soap/envelope/"
>
<S:Body>
   <Employees>
      <Employee>
          <Id>1</Id>
         <Name>Mr. Tester 1</Name>
         <Email/>
         <Fax>7000001</Fax>
         <Phone>8000001</Phone>
         <Mobile/>

<HomePage>www.Testerwebsite1.com</HomePage
>
     </Employee>
   </Employees>
</S:Body>
</S:Envelope>
```

Figure 5: The Proposed Algorithm

There are two main functions in this algorithm, the encoding, and the decoding:

- **Encoding:** (input: string, output: xml stego file)
  1. Read a string character by character
  2. Select a character, and then get the equivalent sequence number from a predefined file. Each number is representing a null value in one of the optional fields
  3. Generate an employee record based on the sequence number
  4. Write the generated employee to the "XML" file

- **Decoding:** (input: blocks[], output: string)
  1. Read the blocks, block by block
  2. Select the block and extract the equivalent number that equals the null tags
  3. Search in our database and select equivalent character to the extracted number
  4. Add the character to the string

3.2 The Algorithm Range

The problem to be solved is to find all the combinations of numbers without repetition and permutation. In order to calculate the possible combinations of blocks that this algorithm can make, the Binomial Coefficient Theory is introduced.

This theory's formula is n!/(n-r)!(r)! assuming that n = number of possible fields, and r is number of elements used in the combination. To calculate how many combinations, we can make out of 5 optional records, so n = 5.

However, each combination can consist of either 1, 2, 3, 4, or 5 elements. This can be considered as the number of spaces as well. Therefore, we should calculate the equation five times where r = 1 → 5 and the summation of these values should be the number of all possible combinations.

Example: Given n = 5, the possible number of combinations that the algorithm can make can be calculated.

$$\frac{5!}{(5-1)!(1)!} + \frac{5!}{(5-2)!(2)!} + \frac{5!}{(5-3)!(3)!} + \frac{5!}{(5-4)!(5)!} + \frac{5!}{(5-5)!(5)!} \quad (1)$$

$$\underline{5 + 10 + 10 + 5\ + 1 = 31\ combinations} \quad (2)$$

The full expression is listed in Table 1:

Table 1: The Algorithm Full Expression

r	Combinations	Total
1	[1],[2],[3],[4],[5]	5
2	[1,2],[1,3],[1,4],[1,5],[2,3],[2,4],[2,5],[3,4],[3,5],[4,5]	10
3	[1,2,3],[1,2,4],[1,2,5],[1,3,4],[1,3,5],[1,4,5],[2,3,4],[2,3,5],[2,4,5],[3,4,5]	10
4	[1,2,3,4],[1,2,3,5],[1,2,4,5],[1,3,4,5],[2,3,4,5]	5
5	[1,2,3,4,5]	1

In case more than these number combinations is needed, the number of records should be increased, which is n. The optimal number of records is seven optional fields, which will give 127 combinations, including all letter cases as well as digits.

3.3 Code Mapping

There should be certain relationship between characters and codes. Each code represents the positions of an element that has the null value or the empty value. For Example, the code "135" means elements 1, 3 and 5 have spaces on them. Table 2 shows an example of mapping for the case of five records.

Table 2: The Code Mapping

Character	Code
a	1
b	2
c	3
d	4
e	5
f	12
g	13
h	14
i	15
j	23
k	24
l	25
m	34
n	35
o	45
p	123
q	1234
r	12345
s	234
t	2345
y	345
v	124
w	125
x	1245
y	134
z	135
	1235

3.4 Pseudo Code And Time Complexity

In **Encoding** functionality, an XML stego file is created based on secret message s.

Input: secret message s
Output: XML stego file

```
1. Read secret message s. --------------- (1)
2. Load XML file (cover file)----------- (1)
3. For each character ci in string s, create new Employee object Ei. ------- (n)
    a. Based on ci, load optional tags in Ei -------- (31) always fixed
4. Save XML file (stego file) ----------- (1)
```

Total complexity = $\vartheta(n)$

In **Decoding** functionality, secret message s is extracted from XML stego file.

Input: XML stego file
Output: secret message s

```
1. Load and parse XML file (stego file) ----- (1)
2. For each Employee element Ei, ----- (n)
    a. Decode secret character ci based on optional tags ---- (31)
3. Construct secret string s from all secret characters (c1 + c2 + .. + ci) --- (1)
4. Return secret message s -------------- (1)
```

Total complexity = $\vartheta(n)$

3.5 Algorithm Advantages

This technique's advantages are as follows:

- Each tag will be located in the right place as the real file. Even when we publish this paper, no reader or a program can confirm that this file is a stego file

- There will be no extra weight in the XML file since the actual data has been used without any additional data

- number of characters is less than one second (refer experiments' section)

- The file size for encoding an enormous number of characters is less than one megabyte (refer experiments' section)

- The exact time for decoding a vast number of blocks is less than one second (refer experiments' section)

3.6 Algorithm Disadvantages

To be fair, the algorithm also has one obvious disadvantage which is the range limitation. As Alrouh's algorithm limitation gives n! combinations, the proposed algorithm will produce less than this value as per the Algorithm Range section. In spite of this, the proposed algorithm is safer and more secure in terms of exposure.

3.7 Implementation and Design

The proposed algorithm is still at the theoretical phase until it can be proven to work. A demo project has been designed to prove the algorithm's validity and efficiency.

Written in Java based on object-oriented design, the main difference between the proposed algorithm and all the reviewed algorithms is that the proposed algorithm will create the data based on the requirements.

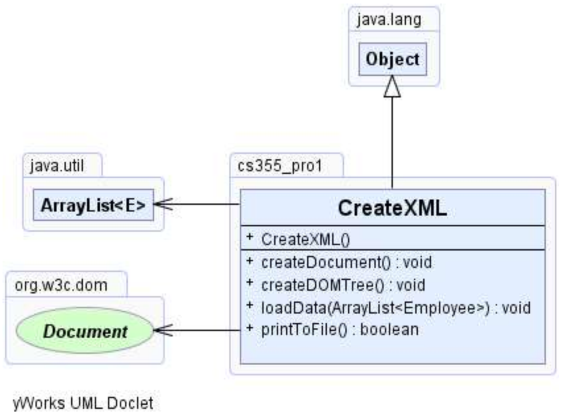


Figure 6: The Class Diagram

The class diagram illustrates all the classes used in this program as well as the relationships between them (Figure 6) with the full class description as follows:

- **Employee:** This class represents the main topic or subject which is employee master data. The most important method in this class is GenEmployee. This method will take the message and convert each character to the equivalent employee based on the mapping file

- **CreateXML:** This class will be used in the encoding process. First, the message to be embedded is to be read. Second, for each character the related code is returned from the mapping file (Table 1). Third, the method GenEmployee is called to create appropriate object of class employee. Finally, the employee data is written in XML file in blocks form (Figure 7)

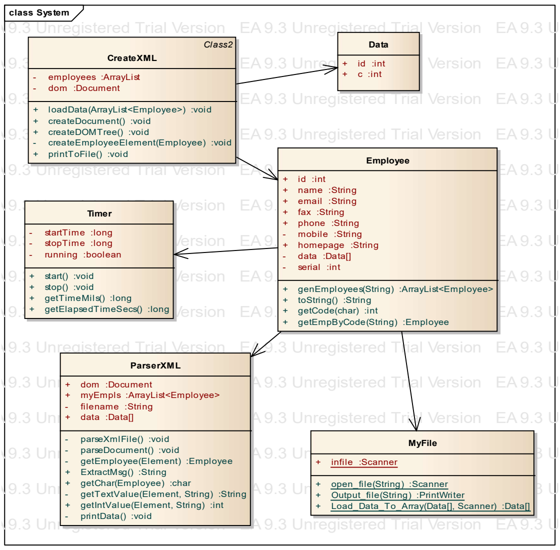


Figure 7: The CreateXML Diagram

- **Data:** This class to be used with an array to form a list of all characters and their related codes. The purpose is to eliminate the time execution needed to access the mapping file (see Table 1)

- **MyFile:** A help class that makes it easier to open and close files

- **ParserXML:** One of the major classes being used in the decoding process. The class will help to convert each block to an employee object. Subsequently, based on this employee data, the proper code will be selected. Finally, the proper character will be selected from mapping file using the code (Figure 8)

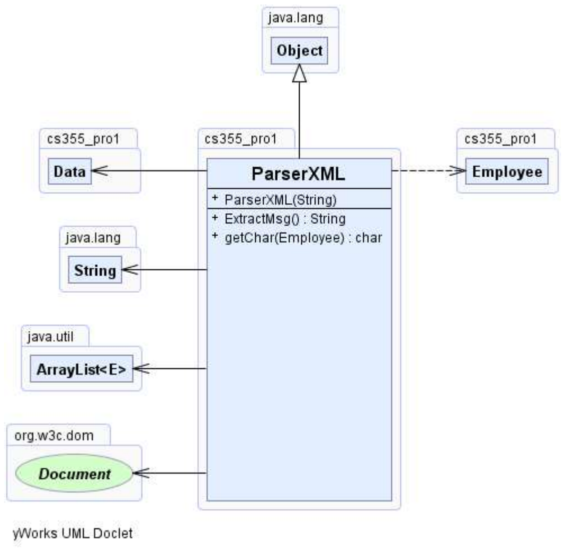


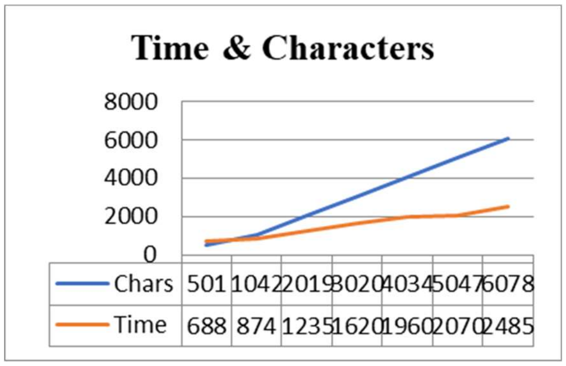
Figure 8: The ParserXML Diagram

4. RESULTS AND DISCUSSION

Experiments to confirm the proposed algorithm's validity and the resulting stego file's efficiency will be discussed. The first comparison will be between execution time in milliseconds (MS) and the number of characters in file size in kilobyte (KB).

The following question will be answered: How much time the program needs to create blocks or employees that are equivalent to a certain number of characters?

4.1 Experiment-1 (Encoding)

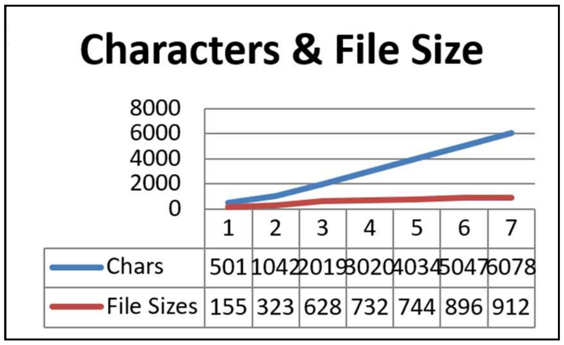


	1	2	3	4	5	6	7
Chars	501	1042	2019	3020	4034	5047	6078
Time	688	874	1235	1620	1960	2070	2485

Figure 9: The Time & Characters Graph

The relationship between a number of characters to be created and the execution time is directly proportional, as well as the curve is polynomial. Also, the program was able to produce 6078 characters in nearly 2.4 seconds which as expected (Figure 9).

4.2 Experiment-2 (Encoding)

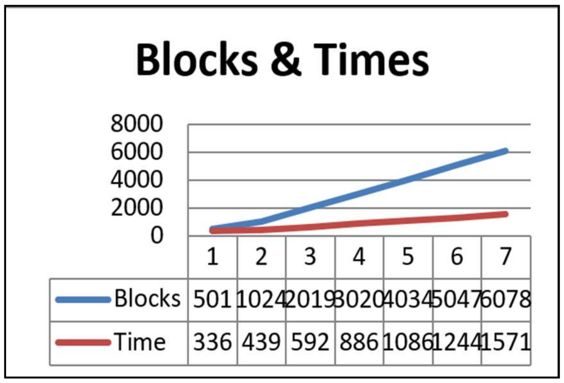


	1	2	3	4	5	6	7
Chars	501	1042	2019	3020	4034	5047	6078
File Sizes	155	323	628	732	744	896	912

Figure 10: The Characters And File Size Graph

The relationship between the number of characters and the file size is reasonable. The program produces 6078 characters, with the file size to be equal to 912 KB, less than 1 MB (Figure 10).

4.3 Experiment-3 (Decoding)



	1	2	3	4	5	6	7
Blocks	501	1024	2019	3020	4034	5047	6078
Time	336	439	592	886	1086	1244	1571

Figure 11: The Blocks & Times Graph

The time to decode the same number of characters is less than the time to encode them due to time to write and create the XML file is added in encoding stage (Figure 11).

The encoding time (Section 4.2) was compared with similar work in literature (Table 3). Aldabagh achieved a much lower encoding time using Least Significant Bit (LSB) as well as his proposed image-and-fish hybrid steganographic technique utilizing image carrier mechanism. However, his message size is only 11K, compared to this paper's proposed algorithm's 912 KB message size, using XML document as the carrier mechanism [25].

In addition, the image file is a single-purpose file restricted to contain only media information and metadata. On the other hand, the XML file, which is embedded with the SOAP tags, can be used to encrypt information for transactions, database records, documentation, as well as a number of other data types. The variety of applications for which the XML file can be put to use provides a huge advantage of this paper's proposed method to steganographically conceal and transport secretive information.

Table 3: Aldabagh's Image-And-Fish Hybrid Algorithm Performance Benchmark

Sample No.	Image Size	Letter Size	Execution Time Original LSB	Execution Time For Proposed Method
1	225 X 225	11K	1.496085 Sec	2.011384 Sec
2	720 X 976	11K	2.106133 Sec	2.099977 Sec
3	200 X 200	11K	1.454461 Sec	2.240116 Sec

For critical analysis, textual steganography endows the greatest challenge in steganographic research primarily based on the fact that extraneous bits are generally not that available compared to image, video, and audio methods.

The risks associated with textual techniques is that differences in textual structure can be definitely get caught by the human eye. On the other hand, image and video structure variations may not be plainly observed. Therefore, it is a tendency to conceal information in images and videos compared to text.

SOAP tags provide an excellent area to camouflage data which is not apparently visible to casual readers. The application of the Binomial Coefficient Theory to minimize the characters combination addresses the issue of text steganography low hiding capacity to an extent [6], comparable to color-coded format-based text method [26], Arabic Kashida font format-based method [27] [28], probability distributions [29] and alphabetic transformation technique [30] in literature.

5. CONCLUSION

Steganography is an emerging field that is potentially able to solve some of cyber security hardest problems. The technique to be used depends mainly on the context together with its own technique.

The author's proposed algorithm makes use of null Email and Mobile tags examples to address existing algorithm restrictions by increasing message size, maintaining the stego file size equivalency to the real file, and minimizing the real message and the stego message differences.

The algorithm encodes and decodes a maximum of 127 character combinations to their corresponding code via code mapping. Since no additional data is used as storage overhead, the algorithm achieved decent performance in three benchmark tests, as well as much better encoding time than similar work in literature.

The experiment makes novel use of SOAP-XML which has not been previously attempted in literature, but it is also applicable to all types of XML files, as it does not depend on the SOAP tags. As XML is widely used for Web-based communications today, this SOAP-XML technique holds a promising solution for a more secure interaction on the Web.

REFRENCES:

[1] Inter-process Communications. Microsoft Support, 2021.

[2] R. J. Reynolds, "Comparison of DCOM and CORBA distributed computing", *Thesis,* New Jersey Institute of Technology (USA), 1999.

[3] "Choosing the Right Remote Object Invocation Protocol in .NET", *Informit*, Pearson Education, Hoboken, NJ (USA), 2003.

[4] H. Abie, "CORBA Firewall Security: Increasing the Security of CORBA Applications", Norwegian Computing Center (Norway), 2000.

[5] B. Li, J. He, J. Huang, Y. Q. Shi, "A survey on image steganography and steganalysis", *Journal of Information Hiding and Multimedia Signal Processing (JIHMSP), Vol* 2, 2011, pp. 142–172.

[6] M. A. Majeed, R. Sulaiman, Z. Shukur, M. K. Hasan, "A Review on Text Steganography

ISSN: **1992-8645** www.jatit.org E-ISSN: **1817-3195**

Techniques", *Mathematics 2021,* Vol. 9, pp. 2829. https://doi.org/10.3390/math9212829.

[7] SOAP Specifications. W3C. https://www.w3.org/TR/soap/.

[8] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, S. Weerawarana, "Unraveling the web services web: An introduction to SOAP, WSDL, and UDDI", *IEEE Internet Computing*, Vol. 6, No. 2, 2000, pp. 86 – 93.

[9] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). https://www.w3.org/TR/soap12-part1/Overview.html

[10] E. -S. M. El-Alfy, A. A. Al-Sadi, "High-capacity image steganography based on overlapped pixel differences and modulus function", *Proceedings of International Conference on Networked Digital Technologies,* Springer Berlin Heidelberg (Germany), Vol. 294, 2012, pp. 243-252.

[11] S. Shakir Baawi, M. R. Mokhtar, R. Sulaiman, "A comparative study on the advancement of text steganography techniques in digital media", *ARPN Journal of Engineering and Applied Sciences,* Vol. 13, No. 5, 2018, pp. 1854-1863.

[12] P. Khare, J. Singh, M. Tiwari, "Digital Image Steganography", *Journal of Engineering Research and Studies,* Vol. 2, No. 3, 2011, pp. 101-104.

[13] K. Rabah, "Steganography - The Art of Hiding Data", *Information Technology Journal*, Vol. 3, No. 3, 2004, pp. 245-269.

[14] V. Sharma, S. Kumar, "A New Approach to Hide Text in Images Using Steganography", *International Journal of Advanced Research in Computer Science and Software Engineering,* Vol. 3, No. 4, 2013, pp. 701-708.

[15] S. El-Seoud, I. Taj-Eddin, "On the Information Hiding Technique Using Least Significant Bits Steganography", *International Journal of Computer Science and Information Security,* Vol. 11, No. 11, 2013, pp. 34-45.

[16] A. A. Bachar Alrouh, "Information Hiding in SOAP Messages: A Steganographic Method for Web Services", *International Journal for Information Security Research (IJISR),* 2011, pp. 61-70.

[17] L.Y. Por, B. Delina, "Information Hiding: A New Approach in Text Steganography", *Proceedings of 7th International Conference on Applied Computer & Applied Computational Science (ACACOS'08),* Hangzhou (China), 2008.

[18] M. Shirali-Shahreza, "Text Steganography by Changing Words Spelling", *Proceedings of 10th International Conference on Advanced Communication Technology (ICACT 2008),* Phoenix Park, (Korea), 2008.

[19] M.H. Shirali-Shahreza, M. Shirali-Shahreza, "A New Synonym Text Steganography", *Proceedings of 4th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2008),* Harbin, (China), 2008.

[20] M. Liu, Y. Guo, L. Zhou, "Text Steganography Based on Online Chat", *Proceedings of Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing,* 2009.

[21] S. Inoue, K. Makino, I. Murase, O. Takizawa, T. Matsumoto, H. Nakagawa, "A Proposal on Information Hiding Methods using XML", *Proceedings of 1st Workshop of NLP and XML,* November, 2001.

[22] A. G. Memon, S. Khawaja, A. Shah, "Steganography: A New Horizon for Safe Communication Through XML", *Journal of Theoretical Information Technology,* Vol. 4, No. 3, 2008, pp. 187-202.

[23] X. Zhang, H. Wang, J. Sun, "An Information Hiding Method based on SOAP", *Proceedings of 3rd International Conference on International Information Hiding and Multimedia Signal Processing (IIH-MSP 2007),* Kaohsiung, (Taiwan), 2007.

[24] B. Alrouh, A. Almohammad, G. Ghinea, "Information Hiding in SOAP Messages: A Steganographic Method for Web Services", *International Journal for Information Security Research (IJISR),* Vol. 1, No. 1, March 2011.

[25] G. M. T. K. Aldabagh, "Proposed Hybrid Algorithm For Image Steganography In Text Security Using Fish Algorithm", *Journal Of Southwest Jiaotong University,* Vol. 54, No. 6, Dec. 2019.

[26] J.K. Sadié, L.M. Metcheka, R. Ndoundam, "A High Capacity Text Steganography Scheme Based On Permutation And Color Coding", arXiv 2020, arXiv:2004.00948.

[27] A. F. Al-Azzawi, "A Multi-Layer Arabic Text Steganographic Method Based On Letter Shaping", *International Journal of Network Security & Its Applications (IJNSA),* Vol. 11. Available online: https://ssrn.com/abstract=3759471, 2019.

[28] A. Taha, A.S. Hammad, M.M. Selim, "A High Capacity Algorithm For Information Hiding In

Arabic Text", *Journal of King Saud University - Computer and Information Sciences,* Vol. 32, 2018, pp. 658–665, [CrossRef].

[29] R. Yang, Z. H. Ling, "Linguistic Steganography by Sampling-based Language Generation", *Proceedings of the 2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC),* Lanzhou, China, 18–21 November, 2019, pp. 1014–1019.

[30] N. Naqvi, A.T. Abbasi, R. Hussain, M. A. Khan, B. Ahmad, "Multilayer Partially Homomorphic Encryption Text Steganography (Mlphe-ts): A Zero-Steganography Approach", *Wireless Personal Communications*, Vol. 103, 2018, pp. 1563–1585. .