Hiding Information in XML Files Using AES Encryption Algorithm

B. Paul[1 *]
[1*]Assistant Professor, Department of Computer Science & Engineering,
Sam Higginbottom University of Agricuture,Technology and Sciences, Allahabad, India
e-mail: binnupaul16@gmail.com

Abstract

In this modern era, the importance of information security has gained a special importance. Steganography techniques can be applied to text file, images, a video file or an audio file. This paper gives a new insight that how Private Key Steganography can be used to give an extra-secure method. This paper gives another dimension of safe correspondence through information hiding on Internet. The experimental results show that the proposed method has high security than others. The suggested method is implemented by using Java language.

Keywords—Steganography, Cryptography, Security,XML, AES Algorithm

1. Introduction

Steganography is derived from the Greek words. “stegos” is termed as “roof or covered” and “graphy” as “writing or drawing”. Thus Steganography is the hidden writing or secret writing. With the help of this, a secret message can be set inside a piece of information and can be sent without being aware of the secret message. In Steganography unauthorized person will be unaware of the secret data being sent by the end user via any carrier such as web page, video file, audio file or image file.
Whereas, in Cryptography the unauthorized person will be aware of the secret data being sent by the end user but since the message will be encrypted by an unknown key (known to only end users), the information cannot be decrypted by him.
The multimedia files are only used as the cover medium but not as a transmission medium. Whereas in web page information hiding process, the web pages can serve as both the cover medium and transmission method to conceal the
secret information.
Extensible Mark-up Language (*XML*) defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. XML is used to store or transport data, while HTML is used to format and display the same data. Whereas these both languages use tags with certain attributes.
In this paper we propose a new approach of information hiding in XML by getting the encrypted message by AES encryption algorithm using a private key and this obtained message will be used as an value for an attribute in XML. Unlike existing approaches in web page, information hiding i.e. changing the case of tags which can be done by changing some letters in the tags, this paper is using an user defined attribute of XML to store the encrypted secret information so that an unintended receiver cannot suspect it as a stego web page. Even if in case unauthorized person suspect, he cannot retrieve the information because of the private key encryption.

Few existing web steganography Techniques are
In the existing models of the web page information concealing methods the researches have been done to conceal the secret information in the tags and attributes of the source code i.e. HTML and XML files [6] as well as the white spaces of the source code [7][8]. The popular techniques used in the existing model are as listed below:

A. Empty tag method

In this process empty tags i.e. either a begin tag quickly taken after by an end tag or an empty tag is used in order to
conceal the secret information. Using these types of empty tags in the source code does not affect the content on the
webpage[9].

B. Line break approach

In this approach by continuously adding the line break tag at the end of each tag the secret information was concealed in
the webpage. Usage of repetitive of line break at the end of each tag does not affect the content of the web page[9].

C. Changing the case of the tag

As the HTML is not a case sensitive language even the changes in the case of the tags may not show any change in the
web page while parsing[10][11].

D. Information hiding based on the attribute value string

The strings used in the attribute values are not case sensitive, so attribute values can be taken for information hiding by keeping the uppercase indicates 1 and lower case indicates 0 [2].

E. Colour code or tag id replacement with Hexadecimal data [12]

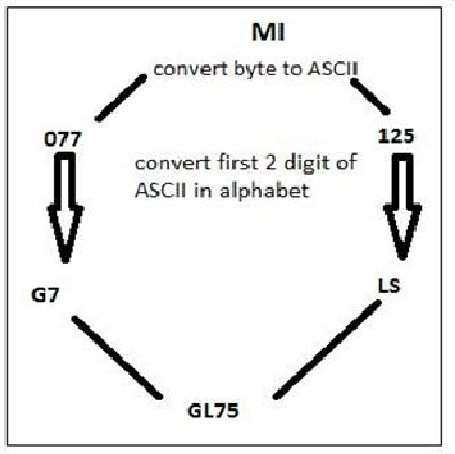

Fig. 1 Tag id replacement

2. METHODOLOGY ADOPTED

One of the attributes that can be used in all the tags in XML. Each attribute(UniqueValue) will be given with a unique identity. The value in this tag attribute will be the encrypted data.

AES ALGORITHM TO ENCRYPT MESSAGE USING PRIVATE KEY[13]

It is based on two common techniques to encrypt and decrypt data knowns as substitution and permutation network (SPN). SPN is a number of mathematical operations that are carried out in block cipher algorithms . AES has the ability to deal with 128 bits (16 bytes) as a fixed plaintext

block size. These 16 bytes are represented in 4x4 matrix and AES operates on a matrix of bytes. In addition, another crucial feature in AES is number of rounds. The number of rounds is relied on the length of key. There are three different key sizes are used by AES algorithm to encrypt and decrypt data such as (128, 192 or 256 bits). The key sizes decide to the number of rounds such as AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys .

In order to create such a cryptosystem, one must remember that anything done by encryption must be undone during decryption, using the same key since it is a conventional (symmetric key) system. Thus the focus is on various invertible operations. One standard technique in using the key is to derive a string somehow from the key, and use **xor** to combine it with the emerging ciphertext. Later the same **xor** reverses this. Otherwise there are ``mixing'' operations that move data around, and ``translation'' (or ``substitution'') operations that replace one piece of data with another. This last operation is usually carried out on small portions of ciphertext using so-called ``S-boxes'', which define replacement strings. One set of mixing, replacements, and exclusive-or with a string derived from the key is called a round. Then there will typically be a number of rounds. The AES uses different numbers of rounds for the different key sizes according to the table below. The table uses a variable **Nb** for the plaintext block size, but it is always **4** words. Originally the AES was going to support different block sizes, but they settled on just one size. However, the AES people (at the NIST) recommend keeping this as a named constant in case a change is ever wanted.

Key Sizes versus Rounds			
	Key Block Size (Nk words)	**Plaintext Block Size (Nb words)**	**Number of Rounds (Nr)**
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Remember that a word is **4** bytes or **32** bits. The names **Nk**, **Nb**, and **Nr** are standard for the AES. In general, I try to use the names in the AES specification, even when they do not conform to Java conventions.

The AES, with its minimum of **128** bits for a key should not be breakable by brute force attacks for a very long time, even with great advances in computer hardware.

Outline of the AES Algorithm.

Here is the AES algorithm is outline form, using Java syntax for the pseudo-code, and much of the AES standard notation:

```
Constants: int Nb = 4; // but it might change someday
        int Nr = 10, 12, or 14; // rounds, for Nk = 4, 6, or 8
Inputs: array in  of 4*Nb bytes // input plaintext
      array out of 4*Nb bytes // output ciphertext
      array w of 4*Nb*(Nr+1) bytes // expanded key
Internal work array:
     state, 2-dim array of 4*Nb bytes, 4 rows and Nb cols
Algorithm:

void Cipher(byte[] in, byte[] out, byte[] w) {
  byte[][] state = new byte[4][Nb];
```

```
  state = in; // actual component-wise copy
  AddRoundKey(state, w, 0, Nb - 1); // see Section 4 below
  for (int round = 1; round < Nr; round++) {
    SubBytes(state); // see Section 3 below
    ShiftRows(state); // see Section 5 below
    MixColumns(state); // see Section 5 below
    AddRoundKey(state, w,
      round*Nb, (round+1)*Nb - 1); // Section 4
  }
  SubBytes(state); // see Section 3 below
  ShiftRows(state); // see Section 5 below
  AddRoundKey(state, w, Nr*Nb, (Nr+1)*Nb - 1); // Section 4
  out = state; // component-wise copy
}
```

Let's go down the items in the above pseudo-code in order:

[1] **Multiplication in GF(256)**: this is not mentioned explicitly above, but the individual functions use it frequently. It is described in Section 2 below.

[2] **Nb**: Right now, this is always **4**, the constant number of **32**-bit words that make up a block for encryption and decryption.

[3] **Nr**: the number of rounds or main iterations of the algorithm. The possible values depend on the three different possible key sizes the earlier table showed.

[4] **in**: the input block of **128** bits of plaintext, arranged as **4*Nb = 16** bytes, numbered **0** to **15** and arranged in sequence.

[5] **out**: the output block of **128** bits of ciphertext, arranged the same as **in**.

[6] **state**: the internal array that is worked on by the AES algorithm. It is arranged as a **4** by **Nb** 2-dimensinal array of bytes (that is, **4** by **4**).

[7] **w**: the expanded key. The initial key is of size **4*Nk** bytes (see table earlier), and this is expanded to the array **w** of **4*Nb*(Nr+1) = 16*(Nr+1)** bytes for input to the encryption algorithm. Each round uses **4*Nb** bytes, and each portion of **w** is used only once. (There is one extra use during sort of an extra round.) This function for expanding the key is described in Section 4 below.

[8] **SubBytes(state)**: this takes each byte of the **state** and independently looks it up in an ``S-box'' table to substitute a different byte for it. (The same S-box table is also used in the key expansion.) Section 3 shows how the S-box table is defined and constructed.

[9] **ShiftRows(state)**: this simply moves around the rows of the **state** array. See Section 5 below.

[10] **MixColumns(state)**: this does a much more complicated mix of the columns of the **state** array. See Section 5 below.

[11] **AddRoundKey(state, w, param1, param2)**: this takes the **4*Nb*(Nr+1)** bytes of the expanded key, **w**, and does an exclusive-or of succesive portions of the expanded key with the changing **state** array. The integer values **param1** and **param2** take on different values during execution of the algorithm, and they give the inclusive range of columns of the expanded key that are used. My implementation of this function doesn't use these parameters, because each round just uses the next **4*Nb** bytes of **w**. The details of this function appear in Section 4 below.

```
<messages>
 <note UniqueValue="Aavgav">
  <to>Tove</to>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
 </note>
 <note UniqueValue="TCZlyo0X">
  <to>Jani</to>
  <heading>Re:Reminder</heading>
  <body>God Bless you</body>
 </note>
<note UniqueValue="0z30pZ+Q==">
  <to>CV</to>
  <heading>Re:Reminder</heading>
  <body>Always be happy.</body>
 </note>
</messages>
```

Fig.2 XML Code

Algorithm for encryption & decryption of data

Input: XML file, Secret Information, Secret key
Output: stego XML file
1. Select the XML file which contains Uniquevalue as attributes in a tag.
2. Input the message and Secret key.
3. Perform encryption using AES encryption algorithm.
4. Separate the encrypted text into different parts.
5. Store all the encrypted code into Uniquevalue attribute of a Tag repository one by one.

Input: stego XML file, Secret key
Output: Secret Information
1. Select the stego XML file which contains encrypted secret message in Uniquevalue as attributes in a tag.
2. Input the Secret key.
3. Get the value from Uniquevalue attributes and concat it with the consecutive values.
4. Perform Decryption using AES Decryption algorithm.
5. Retrieve the secret message which is the Decrypted text.

3. FRAMEWORK OF THE PROPOSED TECHNIQUE

The following diagram depicts the fundamental flow of encryption and decryption process of the proposed technique. Here, we use AES Encryption Algorithm for the Encrypting the data. AES Encryption algorithm uses text message and secret key as an input. And generates the Encrypted message. Decryption process requires the key file and a secret key, as shown in Fig. 3, to obtain the secret message as output.

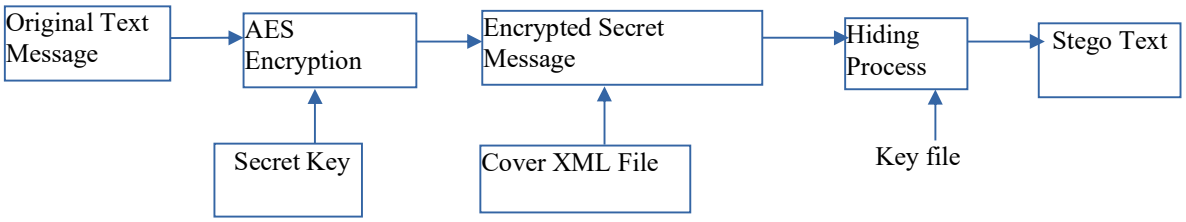

Fig. 3 Data Encryption Process

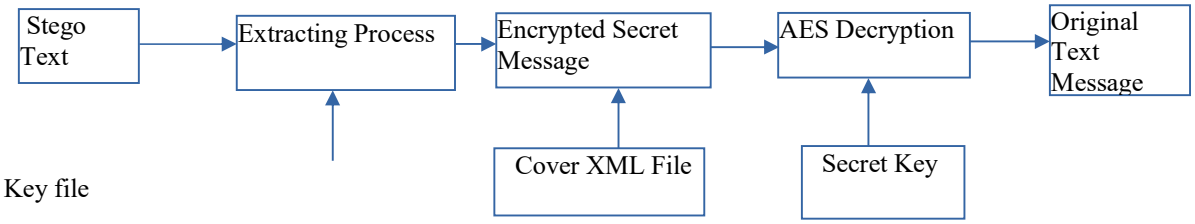

Fig. 4 Data Decryption Process

4. CONCLUSION

Different algorithms have been presented to hide data inside files. Some of these methods were designed to be applied in specific languages, while others can be applied regardless of the language. In this paper, we presented a promising algorithm that can be applied to XML. The proposed method offers high security as compared to other. In addition, this method offers robustness, as the hidden data was inserted inside the in tag attribute as id, and the Internet browser does not show it. Moreover, Using the AES algorithm enhances security by using an encryption mechanism.

REFERENCES

[1] B. Sreekanth Reddy,K.S. Kuppusamy,T. Sivakumar Towards Web Page Steganography with Attribute Truth Table,2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS -2016), Jan. 22 – 23, 2016, Coimbatore, INDIA

[2] X. Yong, L. Juan, and Z. Yilai, “A High Capacity Information Hiding Method for Webpage Based on Tag,” 2012 Third Int. Conf. Digit. Manuf. Autom., pp. 62–65, 2012.

[3] Ram Krishna Singh , Bhavya Alankar,A Novel Approach For Data Hiding In Web Page Steganography Using Encryption With Compression Based Technique,IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 3, Ver. III (May-Jun. 2016)

[4] Chintan Dhanani, Krunal Panchal,”HTML Steganography using Relative links & Multi web-page Embedment” 2014 IJEDR | Volume 2, Issue 2 | ISSN: 2321-9939

[5] Mohit Garg, “A Novel Text steganogrpahy Technique Based on HTML Document”, International Journal of advanced Science and Technology Vol. 35, October 2011.

[6] “Steganography.” [Online]. Available: https://en.wikipedia.org/wiki/Steganography.

[7] Y. C. Chou and H. C. Liao, "A Webpage Data Hiding Method by Using Tag and CSS Attribute Setting," 2014 Tenth Int. Conf. Intell. Inf. Hiding Multimed. Signal Process., pp. 122–125, 2014.

[8] I. Lee and W. Tsai, "Secret Communication through Web Pages Using Special Space Codes in HTML Files," Int. J., pp. 141–149, 2008.

[9] D. V Dhawase and P. S. Chavan, "WEBPAGE INFORMATION HIDING USING PAGE CONTENTS," vol. 3, no. 1, pp. 182–186, 2014.

[10] X. Guo, G. Cheng, C. Zhu, A. Zhou, W. Pan, and D. Truong, "Make Your Webpage Carry Abundant Secret Information Unawarely," 2013 IEEE 10th Int. Conf. High Perform. Comput. Commun. 2013 IEEE Int. Conf. Embed. Ubiquitous Comput., pp. 541–548, 2013.

[11] S. Dey, H. Al-Qaheri, and S. Sanyal, "Embedding Secret Data in Html Web Page," arXiv Prepr. arXiv1004.0459, pp. 1–10, 2010.

[12] Mohammad Shirali Shahreza, "A New Method for Steganogrpahy in HTML Files", Advance in computer, Information and System Science & Engineering , 247-251, 2006 Springer.

[13] The Laws of Cryptography: Introduction to the Advanced Encryption Standard (AES) by Neal R. Wagner
Available:http://www.cs.utsa.edu/~wagner/laws/AESintro.html

[14] Rayarapu, A., AbhinavSaxena, N., & Mundhra, D. (2013). Securing Files Using AES Algorithm.

[15] Shingo Inoue, Ichiro Murase, Osamu Takizawa, Tsutomu Matsumoto, Hiroshi Nakagawa, "A Proposal on information Hiding Mehods Using XML"

[16] Xin-Guang Sui, Hui Luo, "A new Steganography method based on Hypertext", IEEE-2004.

[17] Yujun Yang, Yimei Yang, "An Efficient webpage Information Hiding Method Based on tag Attributes", IEEE-2010.

[18] X. Guo, G. Cheng, C. Zhu, A. Zhou, W. Pan, and D. Truong, "Make Your Webpage Carry Abundant Secret Information Unawarely,"
2013 IEEE 10th Int. Conf. High Perform. Comput. Commun. 2013 IEEE Int. Conf. Embed. Ubiquitous Comput., pp. 541–548, 2013.

[19] S. Dey, H. Al-Qaheri, and S. Sanyal, "Embedding Secret Data in Html Web Page," arXiv Prepr. arXiv1004.0459, pp. 1–10, 2010.