OPTIMIZED HYBRID APPROACH TO SECURE TRANSMITTED MESSAGES USING THE ADVANCED ENCRYPTION STANDARD AND THE WORD SHIFT CODING PROTOCOL

[1]ABDELRAHMAN ALTIGANI, [2]SHIRAZ NASERELDEN

[1]Lecturer.University of Imam Abdulrahman Bin Faisal, College of Science, Mathematics Department, Saudi Arabia

[2]Lecturer. University of Imam Abdulrahman Bin Faisal, Community College - Qatif, Business Administration Department, Saudi Arabia

E-mail: [1]aaaltigani@iau.edu.sa, [2]shnaserelden@iau.edu.sa

ABSTRACT

In [1], a hybrid approach to secure messages using encryption and steganography has been introduced. The methodology of the latter hybrid is to encrypt the secret data using the Advanced Encryption Standard (AES), then hiding the encrypted bits in the spaces of a carrier text using the Word Shift Coding Protocol (WSCP). In WSCP, the hiding process is achieved by increasing or decreasing the font size of the spaces in the carrier text. This hybrid has some limitations. As an extension for [1], these limitations have been resolved.

On the other hand, compared to bare encryption, the performance remains considerably sluggish. For instance, we need 8.87 milliseconds to hide 50 characters using this approach, were we only need 0.12 milliseconds to encrypt the same number of characters using the AES with Counter (CTR) mode of operation.

Keywords: *Textual Steganography, Encryption and Steganography Hybrid, AES, Word Shift Coding Protocol*

1. INTRODUCTION

The convenience and low cost of deploying and using online services has dramatically increased the utilization of E-Services as a complementary or even a replacement for traditional and manual services' ports. It has been reported in [2] that ICT increased the labor productivity by at least 31% in the Europeans Union and by 33% in the united states since 1995.

Unfortunately, a significant segment of the society is still reluctant to utilize E-Services. This is mainly due to the widespread of cybercrimes, or particularly, the cybercrimes which targets consumers (consumer-oriented cybercrimes) [3]. As stated in [4], although the direct effect of cybercrimes is significant, the indirect cost is even higher, as many potential users avoid cyber precarious scenarios by simply avoid using the E-Services. Therefore, unless E-Services have been equipped with proper and reliable security services, E-Services utilization cannot be fully utilized. This includes assuring users that no one can eavesdrop, modify or hinder their transactions. As defined in [5], Cybersecurity is the preservation of the Confidentiality, Integrity and Availability of information in Cyberspace. Hence, we generally need to grant Cybersecurity services to better utilize E-Services.

For instance, to provide the confidentiality security service, we generally use Cryptography or encryption algorithms. Encryption algorithms scrambles the secret data in a way that cannot be easily recovered except for the legitimate receiver.

Encryption algorithms can be classified to symmetric and asymmetric encryption algorithms[1]. The main difference between these two classes is the number of keys required by the algorithm to operate. Symmetric encryption algorithms requires only one key for both encrypting and decrypting payloads. Intuitively, there is a need to share the encryption key securely between communicating parties prior any secure communication. This is the main limitation of symmetric encryption algorithms, which is also known by the "Key Exchange Problem".

On the other hand, asymmetric encryption algorithms resolved this limitation by employing two keys per user, a public key and a private key. These two keys are bound to each other, however it is impossible (given the current feasible computation power) to retrieve the private key from the public key. For instance if the sender (e.g. Alice) wants to send a secure message to the receiver (e.g. Bob), she can fetch Bob's public key, which will generally be accessible for anyone preferably in an integrity protected format. Consequently, she will encrypt the message using Bob's public key. At this point no one, including Alice can decipher the encrypted message. Only the holder of the corresponding private key (i.e. Bob) can decipher the message. Normally, no one except Bob has access to Bob's private key.

The main limitation when using asymmetric encryption algorithms is their poor performance compared to symmetric encryption algorithms. Consequently, most current encryption paradigms incorporate both symmetric and asymmetric encryption algorithms to utilize the good qualities of each. For instance, it is common to encrypt the payload using a symmetric encryption algorithm, then encrypt the symmetric key - used to encrypt the payload - by the recipient's public key. The receiver will decipher the key using his private key, and consequently decipher the message payload using the retrieved key.

When an encrypted message transmitted across a network, anyone in the middle can deduce that a private communication is now taking place between the sender (e.g. Alice) and the receiver (e.g. Bob). Given that a robust encryption algorithm is used, the attacker will not be able to retrieve the content of the encrypted message. In 2001, the National Institute for Standards and Technology has selected rijndael algorithm as the AES. Since then, it has been used by most systems' vendors for encrypting bulks of data.

Under certain circumstances, it might be undesirable to reveal the fact that a secret communication is now taking place. In these cases, steganography algorithms can be a better choice. Steganography is a variant of cryptography. Many definitions exit for the term steganography. For example, in [6], they defined steganography as the art of concealing information by embedding secret information inside an unsuspicious carrier without revealing its presence. Another concise definition for steganography is: "the process of hiding the "existence" of secret message" [7]. Therefore; when using steganography, it might seem like no secret information is hidden at all [8].

We can conclude that, steganography algorithms and techniques tries to hide the existence of the secret information. Therefore, it can be argued that steganography does not provide "real" security, because if a competent attacker suspected that a steganography has been used, he might be able to recover the secret information. Consequently, it has been recommended in [9] to use some sort of a key to increase the robustness of the used steganography algorithm.

Various file types can be used as a carrier to hide the secret information. This includes video, executable (binary), audio, image and text data types. However, text data is the most exchanged data type among casual users. Therefore, using text cover might be a practical choice.

It can be noted that neither bare encryption nor bare steganography fits all scenarios. The decision of whether to use encryption or steganography highly depends on the case requirements. It worth noting that integrating both steganography and encryption might be a better solution in many cases. Therefore, in [1]; a research paper introduced a hybrid that incorporates an encryption algorithm (namely the AES) and a steganography protocol (namely, the word shift coding protocol) to secure the secret message.

The latter model introduced in [1] attracted the interest of many researchers, however it has three main limitations that requires addressing. Therefore, the main objective of the current research paper is to point out and address the limitations of this model in order to make it more convenient and practical.

The next section will review the literature and highlight few other techniques with their limitations. After that, Section three will explore the methodology that has been used in this research.

Consequently, in Section four, some aspects related to the implementation will be tackled. Next to that, in section five, the collected results will be mentioned and discussed. Finally, a conclusion will be provided in section six.

2. BACKGROUND

2.1 Text Steganography

Text steganography algorithms refers to steganography algorithms in which we hide our secret data using text cover. Various text steganography algorithms or techniques has been introduced in the literature. Text steganography techniques can be classified to Semagram, Open Codes and Natural Language Processing techniques [10].

Semagram techniques hide the secret message in the carrier by changing the carrier appearance or adding extra characters to the carrier text [11].

On the other hand, open code techniques rely on sharing a map or a rule that will be used to hide and extract the message between communicating parties. For example, the sender and the receiver can agree to use the second letter of each word in the carrier text to hide and extract the secret message [12]. As a demonstration example for the latter case, the name of the famous writer "Charles Dickens" can be used as a carrier text to hide the secret message "hi". A more sophisticated and realistic example for open codes has been introduced in [13]. In this research paper, the secret message is represented in binary, then each byte (8 bits) is splitted to a pair of 4 bits. The equivalent decimal for both 4 bits is looked up in a "number assignment" table introduced in their paper. This look-up process will return two letters. These two letters can be viewed as the obscured version of the original letter in the secret message. The same process is repeated for all secret message letters. Now, the sender needs to construct a carrier text in which each word must start with a letter from the above obscured set. The process is inverted in the receiver side to extract the secret message [13].

Natural Processing Language techniques such as [14-16] hide the message by manipulating the carrier text grammar, synonyms, words paraphrasing, etc.

2.2 Semagram

This research focuses on semagram steganography algorithms. Therefore, in this subsection we will summarize and discuss some examples for this class of text steganography that has been introduced in the literature.

In [17], they transformed the secret message alphabets and the cover text alphabets to its ASCII binary representation. The bits of the secret message will be hidden over the bits of the cover text. The bits of the cover text are not going to be altered, however the locations of the places in which the secret message has been hidden is going to be appended as additional characters in the cover text.

Obviously, these added characters contain all information required to retrieve the whole message. Anyone in the middle can effectively retrieve the secret message from these additional characters. Hence, this algorithm is not secure enough.

Another approach utilizes a characteristic in Arabic alphabets known as "extensions". In this variant of textual steganography, we need to use an Arabic text cover. In the start we represent the secret message in binary. Whenever a pointed letter like 'Tha' or 'Jeem' appears with extension, this means a bit with the value '1' is hidden. Similarly, if any un-pointed letter such as 'meem', appears with extension, this means a bit of the value '0' is hidden [18].

The key limitation of this steganography protocol is that whenever the opponent notices the extensive use of extensions, and understand that it is used for hiding secret information, the secret data can then be retrieved easily. No further layer of protection has been used to protect the secret message.

Another famous technique is to manipulate ends of lines. This is done by filling ends of lines by spaces or tabs, where the space stands for 1 and the tab stands for 0 (or vice versa). These 0's and 1's composes the secret message [19]. Similar technique called space mimic utilizes the white area at the ends of lines in addition to the white spaces between paragraphs in the same manner described above [20].

To increase the carrier capacity, in [21] they decided to develop a technique that utilizes spaces between words in the cover (inter-word spacing) and spaces between sentences in the cover (inter-line spacing) to hide the secret message. This technique has been called wbStego4open. Spaces are replaced by the Unicode character 0×00 to hide 1 or 0×20 to hide 0. In Microsoft Word file, the values 0×00 and 0×20 cannot be distinguished from the normal space.

One possibility is to utilize the white area between paragraphs (inter-paragraph spacing), and fill it with spaces to hide the secret data. For instance, in [22]; both inter-word spacing and inter-paragraph spacing is utilized. They use Microsoft Word 2007 as carrier. A single space represents a bit in the secret message with the value of 0 and double spaces represents a bit with the value 1.

In [10], they firstly discussed an effective attack called Dot and Arrow Show/Hide (DASH), that can easily reveal data hidden using all steganography approaches introduced in [19-22]. This simple attack utilizes a feature called "show hide formatting" in Microsoft Word. Similar feature is available in other word processors including Open Office. Using this feature, all spaces and tabs will be replaced by "." and "→" characters respectively. Hence, all hidden spaces, double spaces or tabs will be revealed. Consequently, secret data that has been hidden using any of the above four approaches will be jeopardized. After they discussed this vulnerability, they introduced using different Unicode space characters such as En Quad, Em Quad, Thin, etc for the same purpose. These characters will not be replaced with any other character when use "show format" property in Microsoft Word.

The later approach will surely counter the DASH attack. However, if the attacker search explicitly for these Unicode space characters, he will find them, and it will be easy for him to infer that some data are hidden, as these characters are not used in the same frequency normally.

As mentioned previously, in certain scenarios we need to use steganography to hide the fact that a secret communication is now taking place. However, using bare steganography as the only layer of protection might not be the best decision. It has been stated long time ago that the security system must remain secure even if it has fallen in hands of the enemies [23, 24]. Therefore, designing a hybrid algorithm that incorporates both encryption and steganography might be a better alternative for using bare steganography. This will be discussed in the upcoming subsection.

2.3 Integrating Steganography and Encryption Algorithms

Some examples for hybrid algorithms which incorporates both encryption and steganography exist. This means our data will be encrypted prior to applying the steganography scheme. The key idea is that even if the attacker noticed the used steganography, he still cannot retrieve any useful information. This is because the data has been encrypted in the first place.

However, most of hybrids introduced in the literature uses either image or audio file as carrier. For instance, in [25], they addressed the hybrids of cryptography and steganography. They discussed nine different hybrid approaches, seven of them uses image as a cover and the remaining two approaches uses audio files as a cover.

It can be noted that limited research has been conducted on hybrids of text-steganography and Encryption. The authors of this research claims that text files are the most commonly exchanged files for casual users. Therefore, more research is required to introduce convenient schemes of Text-Steganography and Encryption Algorithms hybrids. This research is a step toward this objective.

2.4 WSCP and AES Hybrid

This subsection explores one of the few hybrids of text-steganography and encryption that has been introduced in the literature. In this research [1], they introduced an approach that incorporates both Word Shift Coding Protocol (WSCP) and the AES. In the start, the message will be encrypted using the AES. The encrypted message will be represented in binary. Resulting bits will be hidden in the spaces of the carrier text file. A bit with the value 1 will be hidden in one of the spaces in the cover text by slightly increasing the font size of the corresponding space character. On the other hand, a bit with the value 0 will be hidden by slightly decreasing the font size of the corresponding space character.

The legitimate receiver will extract the encrypted message bits from the spaces, then he will decrypt it to get the plain message. Therefore, if the attacker noticed that the spaces hides some data, still he will not be able to deduce the hidden data, because of the used encryption.

2.5 Limitations on the WSCP and AES Hybrid

2.2.2 Sharing Cryptographic Parameters

In [1], the sender and receiver are expected to share 3 parameters:

- The encryption Key.
- Starting Position (integer).
- Flag (binary string).

The encryption key is used with the encryption algorithm to encrypt the message. The starting position will determine the space number from which we will start hiding the bits of the encrypted message. The flag is the sequence of 0's and 1's that specifies the end of the message. Nothing has been mentioned regarding the mechanism of sharing these parameters with the receiver. Sharing the encryption key is relatively easy. Several approaches to share the encryption key already exist. This includes Diffie-Hellman key exchange algorithm, or use enveloping techniques which encrypt the symmetric key using the receiver's public key and send the encrypted symmetric key to the receiver. However, there is no off the shelf technique that can be used to share the other parameters (i.e. the starting position and the flag).

2.2.3 ECB Mode of Operation

In [1], they used the algorithm AES and the mode of operation Electronic Code Book (ECB) for encryption. However, ECB is not a secure mode of operation [9, 26, 27].

2.2.4 Extract encryption key from password hashing

In [1], they proposed generating the encryption key by hashing the user password. However, it is far easier to retrieve passwords compared to encryption keys. Most users' passwords can be broken using the dictionary attack or guessed using social engineering methods [28]. Ironically, it has been pointed out in [28], that the most used passwords includes the word "password"!

Encryption keys should be generated using a cryptographically sound Pseudo Random Number Generator (PRNG). The seed passed to the PRNG should be as random as possible.

In this research, we introduces an optimized version that addresses all these limitations.

3. METHODOLOGY

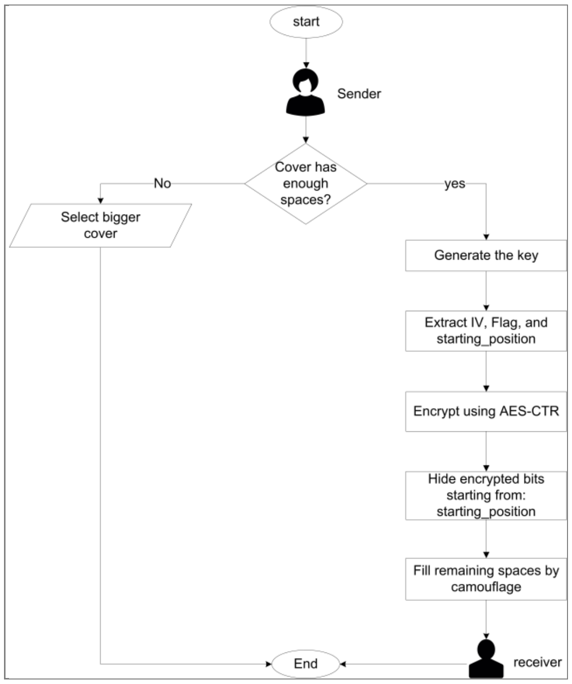
start
Sender
Cover has enough spaces?
No
yes
Select bigger cover
Generate the key
Extract IV, Flag, and starting_position
Encrypt using AES-CTR
Hide encrypted bits starting from: starting_position
Fill remaining spaces by camouflage
End
receiver

Figure 1: Optimized Hybrid Algorithm

Figure 1 illustrates the operation of the optimized hybrid. In the start:

- The sender must ensure that the cover text has a sufficient number of spaces to accommodate all bits of the encrypted message. If letters count in the secret message is n, the count of spaces in the cover text should not be less than $(n\times8)$+starting_position+f, where f is the count of flag bits. The role of the flag and the starting_position will be demonstrated shortly.
- Generate the encryption key (key).
- Extract the Initialization Vector (IV), flag and starting_position as follows:

1. Get the hash function for the key using SHA_256.
2. The 16 left most bytes in SHA_256 (key) will be used as an Initialization Vector (IV).

3. The 16 right most bytes in SHA_256 (key) will be used as a flag.
4. starting_position=((int)(key[31])×(int)(key[30])×(int)(key[29])) mod 100

- Encrypt the secret message using AES algorithm, CTR mode of operation, key and IV.
- Append the flag bits to the encrypted secret message bits. For simplicity, the result of this concatenation will be called the "secret bits".
- In the cover text, starting from the space number (starting_position): hide the secret bits in the spaces of the cover text. The hiding process is performed by slightly increasing or decreasing the font size of the corresponding space character. If the value of the bit to be hidden is 1, then slightly increase the font size of the corresponding space character. Otherwise, if the value of the bit to be hidden is 0, slightly decease the font size of the corresponding space character.
- Randomly, increase or decrease the font size of all remaining spaces in the cover text. Therefore, no one can determine precisely where the bits of the secret message has been hidden, even if he knows that an encryption has been used.

As an example, assume that:

1. The cover text was "This is my cover text that will be used as a carrier text. In this example we just want to show how the model works".
2. Our encrypted message bits where **"10101101"**.
3. The starting position is **2**.
4. The flag is **101**.

- After hiding the secret bits, the cover text will be:"This is my **cover text that will be used as a carrier text. I**n this example we just want to show how the model works".

- After randomly increasing or decreasing the size of remaining spaces in the cover text."**This is my cover text that will be used as a carrier text. In this example we just want to show how the model works**"

- Now the message is secure and ready for transmission.

In this example we magnified the effect to increase the readability, and comprehensibility for the reader. However, in contrast to the above example, the reader is reminded that the increase or decrease in the font size is usually unnoticeable for the naked eye. For example, in Microsoft Word, if the text size is 13, then the font size of the space that hides 1 will be 13.5 and font size of the space that hides 0 will be 12.5.

After the receiver receives the encryption key, he will:

- Extract the IV, starting_position and flag in the same manner described above.
- Extract the secret bits from the spaces of the received text. The retrieval process should start from the space number starting_position. A space with an increased font size represents 1. A space with a decreased font size represents 0. Whenever the receiver finds the flag string, he knows that no more secret bits are hidden.
- Discard the flag bits and decrypt the message.

4. IMPLEMENTATION

The algorithm has been implemented using `C++`. The cryptographic library `CryptoPP` has also been utilized. The carrier is a `MS Word 2007` file.

Inside the `main` function, the routine `encode()` has been be called. `encode()` is a user defined function responsible for both encrypting the secret message (i.e. `plain`) and hiding the ciphered bits of `plain` (i.e. `ciphered`) in the spaces of the cover text (i.e. `cover`) as described above. The result (i.e. encoded text) will be dumped to the file `"encoded.docx"`.

In the start, `encode()` will verify that the number of spaces in `cover` is adequate. In this research we set 99 as a boundary for the value of `starting_position`. The `flag` length is 128, the number of spaces in `cover` should not be less than `(n×8)+227`, where `n` is the number of characters of `plain`. Next to that, inside `encode()`, the routine `get_key()` will be called. This routine will instantiate a `PRNG` of the type `AutoSeededRandomPool`, and uses it to specify the value of `my_key`. After that, we calculate the `SHA256` hash value for `my_key` and read the bytes of the `IV` and the `flag` out of the bytes of the resulting hash using for loops.

Now, our cryptographic parameters are available. So, we perform the actual encryption process using the following lines of code:

```
CTR_Mode<AES>::Encryption e;
e.SetKeyWithIV(my_key,32,IV);
StringSource s(plain, true, new
StreamTransformationFilter(e,new
StringSink(ciphered)));
```

After encrypting `plain`, we determine the `starting_position` and the index of the `flag_begining` space using the following lines of code respectively:

```
int starting_position =
((static_cast<int>(my_key[31]))*(stat
ic_cast<int>(my_key[30]))*(static_cas
t<int>(my_key[29])))%100;
int flag_begining = starting_position
+(ciphered.length())*8;
```

Now, we will hide the bits of `ciphered` on the `cover` spaces starting from the space number `starting_position` up to the space number `flag_begining - 1`. After that, in `cover`, starting from the space number `flag_begining` and up to the space number `flag_begining+127`, we hide the `flag` bits that has been collected as described above. Other spaces (i.e. before `starting_position` or after `flag_begining+127`) will be increased decreased randomly.

In the recipient side, the receiver will call the function `decode()`. `decode()` will extract `starting_position` and `flag` from `my_key`. These parameters will help the receiver to locate the spaces which carries the bits of `ciphered`. After all `ciphered` bits has been collected, the receiver will perform the decryption process.

To assess the performance of the optimized hybrid, we used the function `QueryPerformanceCounter()` before and after calling `encode()` or `decode()`. This function will help in counting the elapsed CPU cycles during executing either `encode()` or `decode()`. The function `QueryPerformanceFrequency()` has also been used to get the CPU frequency (i.e. number of CPU cycles the processor makes in a second). Together, these functions will determine the elapsed time in encoding and decoding secret data using this hybrid. The hybrid performance and other results will be discussed in the upcoming section.

5. RESULTS AND DISCUSSION

This sections analyze and discuss the characteristics of the introduced hybrid algorithm. Particularly, we will discuss the security, practicality and performance of this hybrid.

5.1 Security

As has been mentioned before, in certain scenarios neither bare encryption nor bare steganography is enough. The intruder can hinder the availability of the encrypted traffic. Similarly, skilled intruder can discover and retrieve data hidden using bare steganography.

On the other hand, a message secured using this hybrid has a better chance to survive the availability attacks. This is because the message seems innocent, therefore no good reason for the intruder to interrupt it. Even if the intruder noticed the used steganography, he still cannot retrieve the secret data. Not only because the data has been encrypted using the standard cipher, but also because the no one except communicating parties can identify the spaces in which the secret message has been hidden. The reader is reminded that the values of both starting position (i.e. from where we start hiding our secret bits) and the flag (i.e. the string of bits which represents the end of the message bits) will be as random as the encryption key.

Moreover, compared to [1]:

1. The mode of operation used with the AES cipher (i.e. CTR) is secure (c.f. the unsecure ECB mode of operation).
2. The encryption key will be generated using PRNG. This is more secure than just calculating the hash of a password.

5.2 Practicality

Text might be one of the most frequently exchanged type of files. Hence, text steganography might have a preference over other steganography approaches. Compared to [1], there is no need to share the starting position or the flag. As described in the methodology section, it will be directly extracted from the key.

5.3 Performance.

TABLE 1: INPUT SIZES AND COVER SIZES

Input size	First cover size	Second Cover Size	Third Cover Size
50 characters	800 words (5493 Characters)	900 words (6166 Characters)	1000 words (6832 Characters)
100 characters	1200 words (8187 Characters)	1300 words (8900 Characters)	1400 words (9570 Characters)
200 characters	2000 words (13708 Characters)	2100 words (14378 Characters)	2200 words (15065 Characters)

We compared the performance of the introduced hybrid to bare encryption using AES cipher in CTR mode of operation. The specifications of the machine used are as follows:

- Intel® Core ™ i7-6500 CPU @ 2.50GHz (4 CPUs),~2.6GHz.
- 16 GB RAM.
- Two TB Hard disk.

Three input sizes have been tested: 50, 100 and 200 characters.

The reader is reminded that the cover size affects the performance of the encryption and decryption operations. Hence, with every input size, we inspect the performance of the encryption decryption operations with respect to three cover sizes as illustrated in Table 1.

The input size and the cover sizes that corresponds to this input size has been executed 1,000 times. The average elapsed time for every input size and cover size has been calculated. The summarized results are displayed in Figure 2, Figure 3 and Figure 4. For instance, the average elapsed time to secure 50 characters using this hybrid algorithm when the cover size in 800 words (i.e. 5493 characters) is 8.87 milliseconds. The right most bar column in all subsequent figures (i.e. AES-CTR) represents the time consumed for encrypting 50 characters using AES with the CTR mode of operation. Other results can be interpreted similarly.

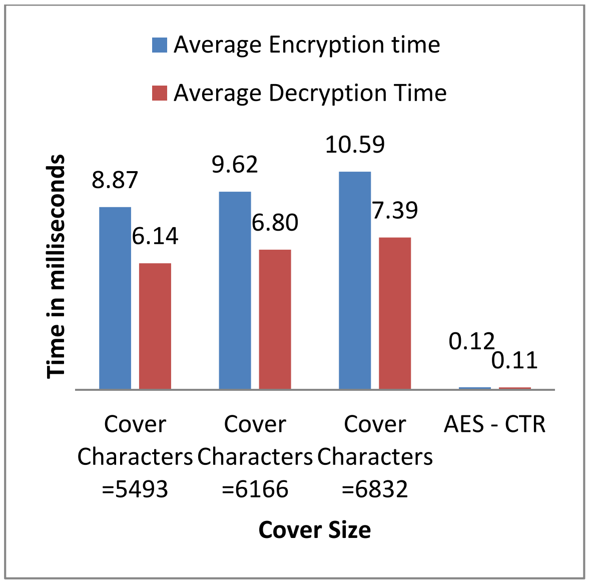

Figure 2: Proposed Algorithm Performance with Input of 50 Characters

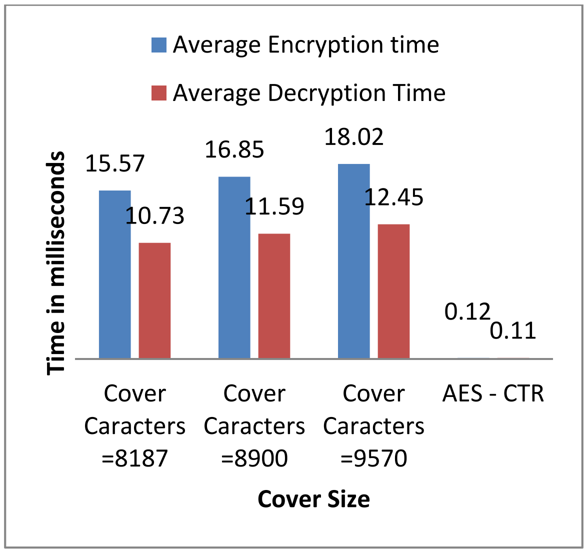

Figure 3: Proposed Algorithm Performance with Input of 100 Characters

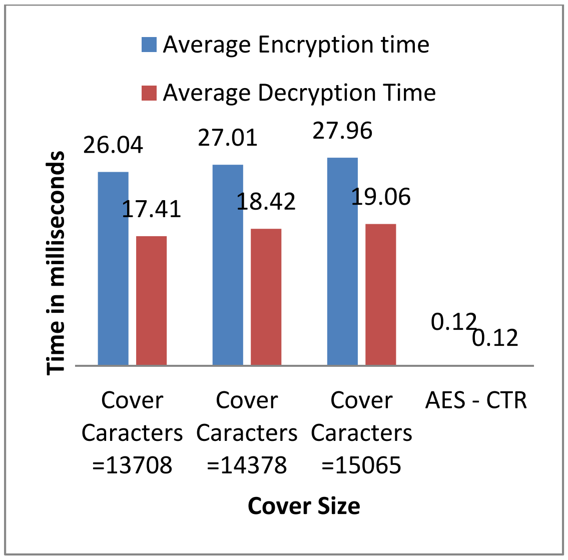

Figure 4: Proposed Algorithm Performance with Input of 200 Characters

The latter three figures show that compared to bare encryption, the performance has been degraded significantly by using this hybrid algorithm. However, the reader should bear in mind that the AES-CTR combination is significantly fast, even if we compared it to other bare ciphers. Moreover, the code used to collect these results is written only to proof the concept. Further code optimizations can significantly increase the performance of the hybrid algorithm. Furthermore, considering the advantage of shielding our encrypted message in additional steganography layer, even the worst reading (i.e. 27.96 millisecond for encrypting 200 characters) might be tolerable. The decision of whether to use this hybrid algorithm or not depends highly on the business requirements and the threat model that we must handle.

5.4 Real World Considerations

Previous points of this section highlighted key contributions of this research. However, further issues must be well considered to grant the practicality of the methodology introduced by this research. The following are some of them:

- **Cover text selection**: The matter of the cover-text selection plays a vital role in the success or failure of this hybrid algorithm. For instance, assume that this model has been used to secure military communications or any other communications of similar sensitivity. In such cases, the cover should be unique every time. Using the same cover repeatedly will probably raise suspicions. Although the attacker cannot attack the confidentiality of the message due-to the encryption layer, he might be able to attack the availability of the message. Similarly, generating random stream of words will not be acceptable for the same above reason. Moreover, we also need to make sure that this cover text is sensible in the used context. For instance, assume that we are using this model to secure military communication. If this traffic is censored, it will be highly suspicious for the observer if the used cover is a shopping list or a romantic poem. In real world implementations, the designer must make sure to properly consider this matter.

- **Performance**: As discussed in section 5.3, the model performance is relatively poor compared to bare encryption. This is due to two main reasons.(1) The additional layer of steganography and, (2) the fact that code optimization was not the main consideration while coding the prototype used to collect above results. The main objective was to proof the concept. However, in real world scenarios performance is vital for casual users. If the performance was poor, users will not use this model regardless what security it has to offer. Hence, when implementing this model for real life applications, code optimization must be well considered.

- **Transparency**: Ideally the user should not use this model explicitly. The user should only use a simple messaging system. The whole process of selecting the cover and embedding the message within this cover should be done behind the scene. The user should not worry about anything rather than sending and receiving messages.

5.5 Potentials and Limitations of the Introduced Model

The introduced model is a hybrid that secure messages using encryption and steganography. This model assumes the following: (1) we are communicating in a hostile environment and, (2) high performance is not a critical requirement. Furthermore, we assume that (3) our traffic is censored by a sophisticated attacker who has enough resources to hinder the availability of the traffic. Therefore we need to make sure that our message looks innocent, so it might reach legitimate receiver. For this purpose, steganography is used. On the other hand we cannot rely solely on steganography. Steganography is effective as a camouflage technique, but for provable security encryption is the standard approach. Using this hybrid will significantly boost the security of the of the messages, as two layers of security has been employed. Moreover, this model uses text files as a carrier. Text files are the most frequently exchanged type of files among casual users. Hence, it is fair to say that the practicality of this model is relatively good. Compared to [1], this model excels in its practicality to share the cryptographic parameters. Moreover compared to [1], better encryption scheme (i.e. AES with CBC mode of operation) has been used. Furthermore, robust PRNG has been used to generate the encryption key. This significantly enhances the strength of the encryption key and consequently the whole model.

On the other hand, this model is sluggish compared to bare encryption. Introducing code optimizations might help in increasing the overall performance, however it might still not be acceptable for some platforms, especially if the performance is a critical requirement and the underlying platform has limited computation resources. Moreover, the matter of automating the "carrier file selection" might require additional programming efforts. For instance, the resulting system might require access to online database of carrier files, or embedding sufficient number of distinct carrier files within the implementation. Both directions might affect the practicality of the model.

6. CONCLUSION

This research introduced an optimization for a previous work which uses both AES and WSCP to secure the message. Optimizations include:

1. More efficient and convenient approach to share the starting position (i.e. the space in the cover from which the hiding process will start) and the flag (the string of zeros and ones which indicate the end of the secret message).
2. Using the CTR mode of operation. CTR is faster and the more secure compared to ECB mode of operation.
3. Using a stronger encryption key, that has been generated using a PRNG.
4. Eliminate the need to share the IV. This step adds little security; however, it adds more convenience to the model.

In addition to strong encryption, this hybrid adds additional two layers of security. One is the steganography layer. Thus, the message might look innocent to the attacker. Consequently, it might have a better chance to reach the intended recipient. The second layer is the obscurity layer. No one except the sender and the legitimate receiver knows which spaces hides the bits of the secret message.

However, the performance of the introduced hybrid might not be acceptable for many applications. Hence, before deciding whether to use this hybrid or not, the reader needs to weigh the gained benefits versus the lost features and decides accordingly.

REFERENCES:

[1] A. Altigani and B. Barry, "A hybrid approach to secure transmitted messages using advanced encryption standard (AES) and Word Shift Coding Protocol," in *Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on*, 2013, pp. 134-139.

[2] M. Cardona, T. Kretschmer, and T. Strobel, "ICT and productivity: conclusions from the empirical literature," *Information Economics and Policy*, vol. 25, pp. 109-125, 2013.

[3] P. Hunton, "The growing phenomenon of crime and the internet: A cybercrime execution and analysis model," *Computer Law & Security Review*, vol. 25, pp. 528-535, 2009.

[4] R. Anderson, C. Barton, R. Böhme, R. Clayton, M. J. Van Eeten, M. Levi, *et al.*, "Measuring the cost of cybercrime," in *The economics of information security and privacy*, ed: Springer, 2013, pp. 265-300.

[5] G. Disterer, "ISO/IEC 27000, 27001 and 27002 for information security management," *Journal of Information Security*, vol. 4, p. 92, 2013.

[6] B. Bajracharya and D. Hua, "Importance of Integrating Cryptography, Steganography, and Digital Watermarking for Undergraduate Curriculum," *CTE Journal*, vol. 5, 2017.

[7] S. Chauhan, J. Kumar, and A. Doegar, "Multiple layer text security using variable block size cryptography and image

steganography," in *Computational Intelligence & Communication Technology (CICT), 2017 3rd International Conference on*, 2017, pp. 1-7.

[8] H. Gupta, R. Gupta, B. Sharma, and S. Gandotra, "Review on Various Techniques of Video Steganography," *Journal of Scientific and Technical Advancements,* vol. 4, pp. 161-164, 2018.

[9] S. William, *Cryptography and network security: principles and practices*: Pearson Education India, 2006.

[10] L. Y. Por, K. Wong, and K. O. Chee, "UniSpaCh: A text-based data hiding method using Unicode space characters," *Journal of Systems and Software,* vol. 85, pp. 1075-1082, 2012.

[11] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM systems journal,* vol. 35, pp. 313-336, 1996.

[12] N. F. Johnson and S. Jajodia, "Exploring steganography: Seeing the unseen," *Computer,* vol. 31, 1998.

[13] S. Roy and P. Venkateswaran, "Online payment system using steganography and visual cryptography," in *Electrical, Electronics and Computer Science (SCEECS), 2014 IEEE Students' Conference on*, 2014, pp. 1-5.

[14] H. Calvo and I. A. Bolshakov, "Using selectional preferences for extending a synonymous paraphrasing method in steganography," *Avances en Ciencias de la Computacion e Ingenieria de Computo-CIC,* pp. 231-242, 2004.

[15] M. Topkara, C. M. Taskiran, and E. J. Delp, "Natural language watermarking," in *Security, Steganography, and Watermarking of Multimedia Contents VII*, 2005, pp. 441-453.

[16] U. Topkara, M. Topkara, and M. J. Atallah, "The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions," in *Proceedings of the 8th workshop on Multimedia and security*, 2006, pp. 164-174.

[17] M. S. S. Iyer and K. Lakhtaria, "New robust and secure alphabet pairing Text Steganography Algorithm," *International Journal of Current Trends in Engineering and Research e-ISSN,* pp. 2455-1392, 2016.

[18] A. Gutub and M. Fattani, "A novel Arabic text steganography method using letter points and extensions," 2007.

[19] M. Kwan. (2013, March 2018). *The SNOW*. Available: http://www.darkside.com.au/snow/

[20] D. McKellar, "Space mimic," 2000.

[21] B. Murphy, "Syntactic information hiding in plain text. Master's thesis," *Computer Science, Trinity College Dublin,* 2001.

[22] L. Y. Por, T. Ang, and B. Delina, "Whitesteg: a new scheme in information hiding using text steganography," *WSEAS Transactions on Computers,* vol. 7, pp. 735-745, 2008.

[23] F. Petitcolas, "La cryptographie militaire," ed, 1883.

[24] C. E. Shannon, "Communication theory of secrecy systems," *Bell Labs Technical Journal,* vol. 28, pp. 656-715, 1949.

[25] A. Baby and H. Krishnan, "Combined Strength of Steganography and Cryptography-A Literature Survey," *International Journal of Advanced Research in Computer Science,* vol. 8, 2017.

[26] A. Altigani, M. Abdelmagid, and B. Barry, "Analyzing the Performance of the Advanced Encryption Standard Block Cipher Modes of Operation: Highlighting the National Institute of Standards and Technology Recommendations," *Indian Journal of Science and Technology,* vol. 9, 2016.

[27] D. Blazhevski, A. Bozhinovski, B. Stojchevska, and V. Pachovski, "Modes of Operation of the AES Algorithm," 2013.

[28] U. Uludag, S. Pankanti, S. Prabhakar, and A. K. Jain, "Biometric cryptosystems: issues and challenges," *Proceedings of the IEEE,* vol. 92, pp. 948-960, 2004.