International Journal on Advanced Science Engineering Information Technology

Vol.11 (2021) No. 4
ISSN: 2088-5334

RSA Over-Encryption Employing RGB Channels through a Steganography Variant

Ismael Martínez[a,*], Walter Fuertes[a], Melany Palacios[a], David Escudero[a], Tatiana Noboa[a]

[a] *Computer Science Department, Universidad de las Fuerzas Armadas ESPE, 17-15-231B, Sangolquí, Ecuador*
*Corresponding author: *iamartinez1@espe.edu.ec*

Abstract— **This study aims to provide a solution for RSA over-encryption based on steganography variant to hide sensitive information that requires security. The current study describes a software program's implementation phases that allow technicians to re-encrypt a message, first encoded using RSA cryptosystems as an extra security layer. Subsequently, this text was converted into an array of bytes stored in Red, Green, and Blue (RGB) layers of one or more generated images, which will finally be sent to the receiver. The programming recursion technique was used to determine the number of images that need to be sent and the bytes corresponding to each of them. These images were created, reorganized, and encrypted according to pre-defined rules. The security of the proposed method relies heavily on the probability theory. Each of the possible patterns that can store data is equally likely to be chosen by an attacker. The program also uses as a conceptual base the Least Significant Bit Substitution (LSB) technique, with the difference that it directly stores bytes for the RGB values of the generated image(s). Our results indicate that the re-encryption is independent of a carrier image. The number of different ways to organize the pixels and the generated images suggests an acceptable security level. Moreover, it requires a not negligible computational power to decipher the pixels' original order and the channels containing relevant information.**

Keywords— **RSA; RGB; over-encryption; steganography; decryption.**

Manuscript received 16 Nov. 2020; revised 12 Jan. 2021; accepted 15 Feb. 2021. Date of publication 31 Aug. 2021.

I. INTRODUCTION

The current study's main aim has been to provide a novel method that allows a transmitter to conduct a coded message. The intention of the given message remains hidden and remains only understandable for its recipient. To carry out, we implemented an algorithm to hide information on a set of generated images by firstly encrypting text with a variant of the already known RSA algorithm. Later, we may use the encrypted text's obtained data to create one or several images whose pixels contain pieces of data that may be reorganized in any desired order and subsequently sent to the receiver. More specifically, after applying the baseline RSA algorithm [1] to a message on the network, this resulting string will be stored in one or several images depending on the length of the message.

There has been some research considering the over-encryption and data hiding in digital images. Recent studies of Liu and Zhang [2] aimed to provide an improved version of the Least Significant Bit (LSB) algorithm to hide information. They combined the steganography and QR code (abbreviated from Quick Response, QR) to hide an encrypted image by storing the last most significant bits in the carrier image is different red, green, and blue (RGB) channels. The study proposed by Mukherjee *et al.*[3] encrypts an image using a variant of the RSA algorithm, separating the image into several files. Nath [4] developed multiple encryption methods based on Steganography using an encrypted secret message. Khan [5] proposes a steganographic method based on a grey-level modification for authentic color images using image transposition, secret key, and cryptography. Sultan \and Al-Shaaby [6] conducted a comparative study of steganography and cryptography. They surveyed some methods combining cryptography and steganography techniques in one system. Somyia *et al.* [7] proposed a new data hiding approach, which utilizes some AES data encryption concepts while hiding the data using hex symbols steganography. For all of those mentioned above, the joining of cryptography and steganography helps reach a near-perfect communication system, presenting higher reliability than stand-alone cryptographic methods. The joining of RSA and steganography builds a robust way to encrypt capable of multiple front types of attacks, detection, and reverse engineering.

Another use case of steganography showcased by Gutub and Alaseri [8] demonstrates this technique's usage within Arabic text, taking as a foundation a single character to hide computer-generated passwords within user-selected passwords. Several prior proposals combine traditional steganography to hide information within multimedia files with cryptography techniques [9]–[15] to encrypt the information or encrypt the whole stage-medium carrying data. Studies using asymmetric cryptosystems such as RSA and steganography [16]–[18] had been proposed in the past. Nevertheless, the proposed methods use the LSB technique to hide data, making them dependent on a carrier file.

All these efforts have provided various solutions, which also shows the interest of the scientific community. However, the vulnerability of some versions of the encryption algorithms persists. Thus, we explore a new way of storing data and taking advantage of the RGB channels available for each pixel on an image, focusing on strengthening the encrypted text.

The current study describes the implementation phases of a procedure that allows us to convert an encrypted text into a series of generated images sent to a receiver within this context. Recursion has been used to determine the number of images sent and the bytes that correspond to each image. Computers generate these images, and their pixels may be reorganized and encrypted according to a set of pre-defined rules. Through this work, a new alternative is proposed to store information in the RGB channels of an image generated from a message, keeping the location secret and channels containing relevant information when decrypting the resulting image.

The main contribution is the proposal of an algorithm that allows us to transform a message into a set of images, performing as a variant of the already known steganography technique. The proposed algorithm does not depend on any carrier, as usually, typical steganography does. This algorithm generates its own set of images whose RGB layers (i.e., the primary colors in additive color synthesis) contain pieces of data and are organized using pre-defined rules.

The finished results demonstrate that the over-encryption remains independent of a carrier image. Furthermore, the number of different organizing pixels and generated images allows the proposed encryption method to appear very secure. In attempting to decrypt the message without the knowledge of the mentioned process, it may require an enormous and unrelated amount of computational power to uncover the original order of the pixels and the channels that contain relevant information. Equally important, during the development of this research, we verified that the size and weight of the created images have been considerably smaller than those of the images used in stenography.

The rest of this paper is structured as follows: Section 2 presents Materials and Methods that explain all the procedures used to conduct this research and the undertaken methods. Section 3 provides the evaluation of results, the complexity analysis of algorithms, and the Discussion of findings. Finally, Section 4 presents the conclusions as well as future work lines.

II. Material and Method

A. *Requirements Specification*

According to the IEEE Software Requirements specification [19] is the arrangement for a particular software product, program, or set of applications that perform certain functions in a specific environment, intending to provide an adequate solution. Within the context of the current study, the following scenarios have conceived:

The security of the RSA algorithm depends on the classical difficulty of factoring large integers [3]. Furthermore, the security is affected by the decomposition into prime factors, which for more reliable security requires a greater length of the key, implying an increase in computational costs [20]. Additionally, when small values are selected for the key, the encryption process remains weak, and the security decreases, causing easier decryption of the message [21].

As stated in Alia *et al.* [22], "steganography is defined as the art of hiding secret data in a non-secret digital carrier called cover media". Cryptography focuses on keeping the contents of a message secret, while steganography focuses on keeping the existence of that message secret. Cryptography encrypts a message, but steganography hides it. However, as stated in Ahmad [23], the proposed steganography solutions so far have a limitation on the embedding capacity for colored images without affecting the images' quality.

For the reasons mentioned above, a variant of the basic steganography has been adopted in this study. Thus, it has been considered that to perform encryption to become more challenging to infringe, the RSA encryption algorithm needs to be applied, which is based on selecting two large prime numbers chosen at random and kept secret. This algorithm's main advantage of security rests in the difficulty when factoring in large numbers [20]. In this case, it is used to encrypt a message that will then be sent through an encoded image.

B. *Model Design*

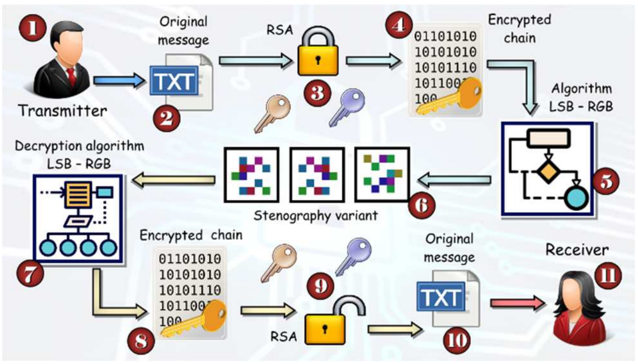


Fig. 1 Schematic illustration of the experimental model which has been tested in a controlled virtual network environment

We have tried to improve the process of over-encrypting a message [24] to send it over the Web, by encrypting the RSA algorithm's result, using a variant of steganography (i.e., over-encryption). The procedure is as follows (see Fig.1). At (1), a transmitter chooses to use this method to hide data. At (2), the transmitter will proceed to write a message that will be known as the original message. At (3), this chain will be introduced to an encryption process using the RSA algorithm that has already been described. At (4), subsequently, as a result, we will obtain an encrypted chain. At (5), once we have this output, it will be decomposed into bytes, storing them into an array. Then at (6), to create the images, this first array needs to be separated into various arrays, containing a maximum of 255 bytes. Once the hiding method has been completed, images with the maximum number of bytes containing relevant data hidden on specific channels, layers, and sequences will be obtained. At (7), while these images arrive at the other computer, the process of retrieving the information begins. The bytes of data hidden on the image must be retrieved and saved on an array. Therefore, the algorithm will continue to add data extracted from each image to this particular array. Later at (8), the end of the process will result in a group of bytes representing an encrypted chain. At (9), this chain is rebuilt and decrypted using the RSA algorithm. At (10), the original message is presented to the receiver, as shown at (11).

C. Model Explanation

The model for the proposed data hiding method in images is described below. It will hide encrypted text bytes inside 24 bits of colors automatically generated images using the encrypted text's bytes using RSA. The image that contains the hidden data will be saved as a BMP format with no data loss caused by the compression other formats offer. We refer to byte or bytes for eight bits representing the encrypted message in decimal notation, in which the obtained values range from 0 to 255.

1) Data Hiding: When it comes to data hiding, the process begins retrieving all the bytes from the encrypted text, as these values are integer numbers higher than zero. As previously mentioned, we will use them as pure colors for each channel of the generated image using data from the encrypted text. As the proposed method hides only one byte of the encrypted message per channel on each pixel, Table 1 (a) indicates all possible data hiding options. However, this table contains some useless possibilities for the proposed method as it indicates combinations that use only one or none channels to store data.

Table 1 (b) may report all the combinations that are useful for the method mentioned above. Disclaimer: 1s and 0s only represent locations where relevant data will be hidden. A brief description of the procedure is given below:

TABLE I
(A) BLENDS IN THE RGB LAYERS. (B) USEFUL COMBINATIONS FOR THE METHOD

(a)

R	G	B
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

(b)

R	G	B
1	0	0
1	0	1
1	1	0
1	1	1

2) Establishing the Dimensions of the Generated Images: The number of pixels that compounds the generated images will depend on the number of bytes retrieved from the encrypted message. The resulting number will subsequently be divided by two, as the proposed method uses two channels per pixel to store relevant data. However, it is implausible to receive an image with fractions of pixels; this means that if the value obtained from the last division is a fractional number, it needs to be approximated to the closest superior integer. In this way, we prevent running out of pixels to store data. Once the number of pixels has been determined, we must obtain its square root, as the proposed method creates square images with equal width and height. As previously mentioned, if the number obtained is fractional, it needs to be approximated to the closest superior integer.

3) Storing the Real Quantity of Data being Hidden on the Image: In most cases, our method requires approximations to create images with the same width and height, causing an alteration of the real number of data is being hidden. Therefore, there must be one pixel and channel containing the real number of bytes stored on the image pixels in the resulting images. This pixel demands to be the first one to be located before the retrieving of data starts. Due to the images of 8-bit color depth, the proposed method creates x number of images, of which each one contains a maximum of 255 bytes of relevant data.

4) Storing the Bytes Retrieved from the Encrypted Message: For the sake of simplicity, the way the bytes are hidden appears in the following form. The algorithm will start at the first row and column of the image, as it is treated as a two-dimensional matrix. According to the established rules, the algorithm will choose the pixel channels that are being treated to save the data. After such a process is completed for every image generated by the algorithm, each image's pixels and channels may be shuffled independently of the other images using other algorithms, as documented in this study's related works.

5) Sending Images: Once the hiding process is complete, the images need to be sent and stored in the order in which they were originally created. The algorithm implements one function that stores them in the mentioned order, as it is called each time the hiding process for each image has ended (see pseudocode 1).

Algorithm 1 Pseudocode: Sending Images

```
1:   procedure STORE THE IMAGE
2:   Begin:
3:     For each image received
4:     If the image needs to be encrypted;
5:     A new File is added;
6:     Message, the String terminal and the encrypted text is
       added;
7:     If the image needs to be decrypted;
8:     A new File is added;
9:     Message, the String terminal and the decrypted text is
       added;
10:    The mistakes are verified;
11:  End.
```

6) Data Retrieving: When a batch of images is received, retrieving data will start automatically. If the pixels and channels have been shuffled on the hiding process, they need to be returned to the original position. This action needs to be done so that the message that has been hidden is retrieved unaltered. The first step that needs to be completed for the algorithm to continue retrieving data is locating the pixel that contains the real number of data that has been hidden in the image. The following steps are simple: the algorithm will start again from the first column and row of the image, passing over each pixel, the algorithm will check its position. For example, suppose the row and column are odd numbers. In that case, it will retrieve relevant data from specific channels in the order there were saved, ensuring that the message is not modified in any way, according to the specified rules. The bytes retrieved from one image need to be stored to keep adding other bytes that will be retrieved from the other images. When all the bytes have been extracted from each image, it is necessary to build a string using them. Next, the resulting string needs to be decrypted using the RSA algorithm to determine the message's real content.

D. *Security Level of the Algorithm Based on the Theory of Probability*

The security of this algorithm is concentrated in two steps. Firstly, in the number of possible ways in which pixels may be ordered in an image and secondly in all imaginable forms of ordering the bytes in each pixel's three-color layers. An example of such a procedure may serve the following case: we may take a picture of five by five pixels long and wide, which results in a total of twenty-five pixels. Understanding the importance of the order of the pixels in an image becomes more explicit when swapping the twenty-five pixels will result in 1.551121004 1025 different ways in which pixels may be sorted, i.e., different images. Furthermore, this value needs to be multiplied by three, as it represents the number of possible ways in which information may be saved on each pixel when using two layers to store data.

In general terms, it is possible to calculate the number of images that can be created using Equation (1). Where, nP is the number of pixels, and N is the number of images created.

$$N = 3 * (nP!) \tag{1}$$

It is also possible to determine the image's security if it is known that each permutation is equally likely to be chosen by an attacker, based on the probability theory [25]. Therefore, a probability formula may be determined in which the attacker can guess the correct order of the pixels, as indicated by the equation (2).

$$\mathrm{P} = \frac{1}{(3*(\mathrm{nP!}))} \tag{2}$$

E. *Algorithm to Encrypt a Message using RGB in an Image*

The algorithm designed uses as information the bytes obtained from a message, which RSA has previously encrypted. As these values will be positive integers, they will be saved in an image using the RGB format's color layers. In each pixel of the image, it will be possible to save three bytes of the encrypted message (see pseudocode 2).

Algorithm 2 Pseudocode: Multiple-Encryption algorithm

```
1:    procedure MULTIPLE-ENCRYPTION
2:    Begin:
3:       Encrypt the message using RSA;
4:       Store in an array the bytes obtained from the encrypted
         message;
5:       Do
6:       Separate the resulting array into two different arrays
7:       Add the first one to an ordered list
8:       While bytes great than 255 do
9:       For each byte in the list
10:      a. Calculates dimensions of the image;
11:      b. Creates the image;
12:      c. for each pixel of the created image save information
            on each channel;
13:      d. Save the image in a list
14:      Add the first one to an ordered list
15:   End
```

F. *The dimension of the Image Created by the algorithm*

The number of pixels that the image will contain will be defined by the number of bytes obtained from the encrypted message divided by two. However, it is improbable to have pixel fractions in an image. Therefore, the previously reached value must be approximated to the nearest higher integer to obtain a decimal value, or it may risk the loss of data due to the lack of storage space. However, the image dimensions are defined as the square root of the number of pixels that will have the image, since the aim has been the creation of square images. Again, if decimal values are obtained, it will approximate them to the nearest integer value.

Disclaimer: It is fundamental to consider when the algorithm complexity has been based on the number of entries. Note that the original message has been placed in a matrix with an almost equal number of processed entries. Only a few augmented values reach their size to be linearly proportional to the original and not affected by the usage of two nested loops, suggesting a quadratic augmentation of the size.

At this point, it might be implied that to obtain the original number of data containing the image, one would raise the wide and high to the square and multiply it by two. However, when obtaining the image's dimensions, it loses the exact number of data that may have been there in some cases. Therefore, it is considered to reserve a color layer of a specific pixel to store the real value of the number of data contained in the image, allowing a more straightforward decryption process.

G. *Information by Color Layer*

Hiding information in images through the RGB color layers has been used in several works related to steganography. Liu and Zhang [2], like Abbas *et al* [26] used this concept in their studies through the LSB method. The proposed algorithm uses the basis of this concept, differing in that it stores bytes directly for the RGB values of the created image(s). When storing the bytes in each color layer, utmost care needs to be considered. Observing that the values of the bytes are saved from the allowed range for the colors, means that the value needs to be between 0 and 255 since it is established that a layer must be reserved of a specific pixel to store the amount of data that will contain the image (critical data for encryption). If this number exceeds the limit, it will create the number of images needed to store the entire message.

This process is performed using recursion to separate the original array of bytes into n fixes with a maximum of 255 data to create n images later on. In this way, if, after encrypting the message, it is found that the information byte, for example, is 525, it is needed to create three images, of which two of them will contain 255 data and the third some 15. The images created to hide the message are stored in an ordered list sent to the receiver.

H. Decryption Process

The decryption process follows the defined rules in the encryption algorithm and related to those used to identify how the pixel and RGB layer of the image relay useful information. The decryption algorithm represents exactly the inverse process, where the received list of images is considered. As mentioned above, the rules are applied image by image to obtain the byte array stored in them and, later, decrypt using the algorithm of the decryption RSA baseline. The decryption algorithm, represented in pseudocode 3, is described below.

Algorithm 3 Pseudocode: Decryption algorithm

```
1:  procedure OVER-DECRYPTION
2:  Begin:
3:          For each image received
4:          a. For each pixel of the image;
5:          a.1. Obtain the values of Red, Green and Blue;
6:          a.1.1 Retrieve information from each channel
7:          a.2. Save the recovered information in an array;
8:          b. Store the array created in a list;
9:          Join all the arrays of the list mentioned above into one;
10:         Create a string using the bytes of the previously created
            array;
11:         Decrypt the string using RSA baseline;
12: End
```

I. Experimental Topology and Proof of Concept

As a test topology, a connection between two computers has been established in two different geographical locations: Valle de Los Chillos and Quito, both situated in the Inter-Andean Valley of Ecuador. For research purposes, a controlled Virtual Network Environment (VNE) has been used. According to Fuertes *et al.* [27], a VNE can be defined as "a set of virtual equipment connected collectively in a given topology, which emulates an equivalent LAN/WAN in which the environment is perceived as if it were real". The sending and receiving of messages of different lengths have been tested, verifying that the data lacked loss and that the decrypted message identical to the original.

When sending the message, the text is encrypted with RSA, and later hidden in n images containing the bytes. If an entity is trying to find the messages' content, it will find a group of images travelling on the Web. The bytes of relevant information of each image needs to be recovered when receiving the group of images. Finally, it may build a message out of them that will be decrypted using RSA by displaying the original text on the receiver screen, as shown in Fig. 2.

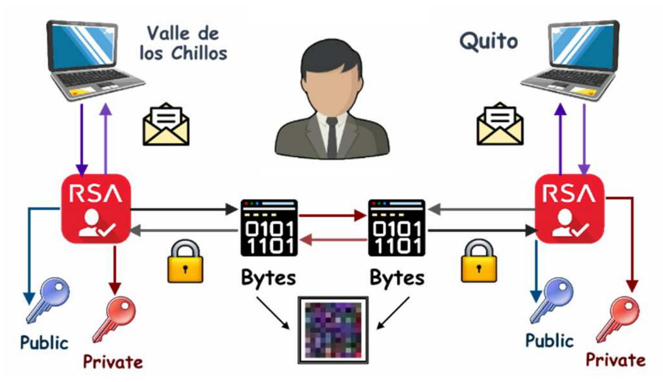


Fig. 2 Encryption and Decryption Process

III. Results and Discussion

The following examples are obtained after implementing a simple JAVA application, which allows an individual to write a phrase and subsequently encrypt it. An overview of the RSA encryption process result is also detailed, as it is fundamental to understand how the algorithm works. The string of characters is the input for the encryption algorithm. It is lately encrypted and hidden in the images created by the algorithm which outputs a batch of images sent through the internet. It also illustrates the image or images resulting from the encryption process so that the difference between standard steganography and this variant becomes visible.

A. Example with a Single Resulting Image

When entering the next string of characters: *This is a hidden message sent through an image*, the following RSA encryption obtained is:
4958 3768 790 102 4584 790 102 4584 2467 4584 3768 790 3350 3350 1881 3869 4584 6069 1881 102 102 2467 4229 1881 4584 102 1881 3869 4958 4584 4958 3768 6317 3432 3210 4229 3768 4584 2467 3869 4584 790 6069 2467 4229 1881

The above message encrypted in RSA is converted to a byte array with a length of 221. This length is divided by two, which is approximated to the nearest superior integer, in this example, 110. From the square root of that number, one obtains the image dimensions, which are 11 high by 11 high and wide. Once the image is created through measurement, the rules are applied to locate the bytes in the different RGB channels. The process mentioned above, and all steps are illustrated in Figure 3.

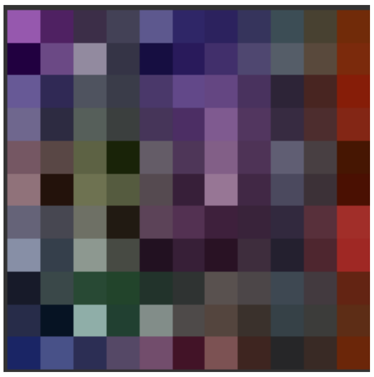


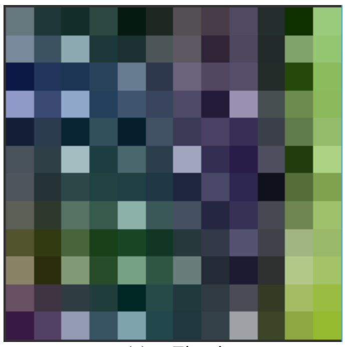
Fig. 3 Image generated with the algorithm

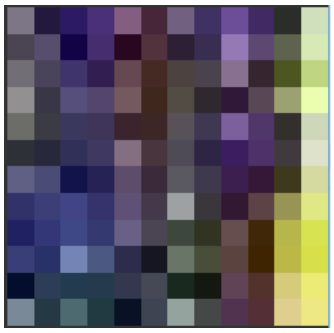
B. *Example with Several Resulting Images*

In order to visualize the results in longer messages, the following text has been considered: The present work seeks to protect sensitive information through the use of the RSA encryption technique and a variant of the steganography technique. The following RSA encryption is obtained:

4966 6724 3768 1881 4584 2677 6317 1881 102 1881 3869 4958 4584 3146 3432 6317 1404 4584 102 1881 1881 1404 102 4584 4958 3432 4584 2677 6317 3432 4958 1881 1851 4958 4584 102 1881 3869 102 790 4958 790 1745 1881 4584 790 3869 2117 3432 6317 6069 2467 4958 790 3432 3869 4584 4958 3768 6317 3432 3210 4229 3768 4584 4958 3768 1881 4584 3210 102 1881 4584 3432 2117 4584
4958 3768 1881 4584 1991 3735 6004 4584 1881 3869 1851 6317 3249 2677 4958 790 3432 3869 4584 4958 1881 1851 3768 3869 790 1027 3210 1881 4584 2467 3869 3350 4584 2467 4584 1745 2467 6317 790 2467 3869 4958 4584 3432 2117 4584 4958 3768 1881 4584 102 4958 1881 4229 2467 3869 3432 4229 6317 2467 2677 3768 3249 4584 4958 1881 1851 3768 3869 790 1027 3210 1881 3143

By decomposing the resulting encrypted message, an array of 725 bytes long has been obtained. Therefore, it has been necessary to divide the message into three images, of which the first image contains 255 bytes, and its dimension area is 12 by 12 high and wide. Similarly, the second image has the same characteristics as the first one as the length of its bytes is about 255. Finally, the third image contains 215 bytes, and its dimensions are 11 by 11 high and wide. Once the images have been defined, the bytes are hidden in the RGB layers. The resulting images have been illustrated in Figure 4.


(a) First image


(b) Second image

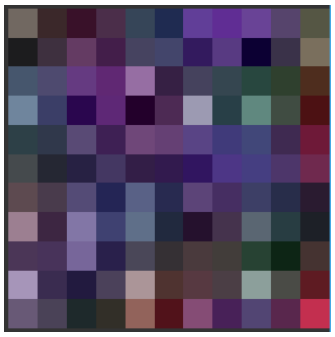

(c) Third image

Fig. 4 Three images generated with the applied algorithm

C. *Time Measurements*

The data in Table 2 compare the encryption and decryption time using different message lengths.

TABLE II
TIME MEASUREMENTS

Number of characters	RSA based encryption	RSA base decryption	Encryption time (s)	Decryption time (s)
104	0.18	0.26	0.18	0.3
208	0.27	0.65	0.31	0.66
366	0.41	0.76	0.46	0.85
539	0.59	0.92	0.63	0.98
733	0.76	1.29	1.05	1.79

D. *Comparison According to the Weight of the Image*

Table 3 lists encrypted images with the algorithm proposed in this work compared to the basic steganography algorithm, similar to that described in Abbas *et al.* [26]. Significant differences in the weights result from the given images. Some encrypted images with basic steganography have been created to demonstrate the comparison. The obtained images are illustrated in Figure 5.

(a) First image

(b) Second image

Fig. 5 Encrypted images with basic steganography

TABLE III
COMPARISON ACCORDING TO THE WEIGHT OF THE IMAGE

Text to encrypt	Encrypted image: proposed method	Weight (Bytes)	Encrypted image: steganography	Weight (Bytes)
This is a hidden message sent through an image	Figure 3	725	Figure 5a	246000
This is through the use of the RSA encryption algorithm and a variant of the steganography technique	Figure 4	2095	Figure 5b	248000

E. *Complexity analysis*

In this section, the characterization of the run time of the algorithm has been conducted, considering the size of the data structure provided, as the original input (character string in natural language). Besides, an analysis of the space used for data processing will also be performed, as explained further below:

The string generated after the original message has been encrypted (i.e., original string) through the RSA algorithm and will be transformed into bytes (i.e., data type handled by programming languages). These bytes oscillate in values between 0 and 255, coinciding with the number of values used to describe the RGB color standard. The growth rate of computational steps has been analyzed as they depend on the input size (i.e., the original string characters). The equivalent is converted to an approximate average of five bytes per character when given the RSA conversion process. The number of entries being the number of bytes resulting from the previous process.

In this way, we define the following variables: n is the number of characters of the original string; x is the number of digits obtained after encrypting the original string using RSA (those that are treated as characters), and y is the number of bytes that are obtained after decomposing the encrypted string for each x. From the above, the number of bytes to process resulting from a string of n characters may be defined by the equation (3).

$$x = 4 * n \tag{3}$$

$$y = 5 * x$$

This algorithm demonstrates a growth complexity O(n). The algorithm's nature will always move an array of constant dimensions through the encryption process. That will be the number of performed operations, which will not increase with each completed cycle. When setting the image's dimensions, the total input number is divided by two and later, get its square root to establish the dimension of each side of the matrix. Therefore, when traversing the matrix through nested loops, we have a total of operations defined by the multiplication of the image's height and width. However, being the same, we obtain the constant value of n/2 operations per image.

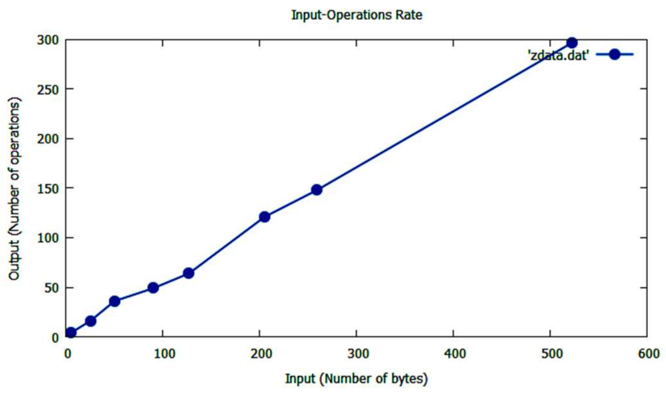


Fig. 6 The graphic demonstration of the iterations growth rate by stretching the number of iterations versus the input in bytes.

Nevertheless, it is fundamental to emphasize that it may be very likely to need more than a single image in a normal process of hiding a message. Therefore, it will be necessary to add every performed n/2 operation about each image to receive the algorithm's total operations. As illustrated in Figure 6, the number of bytes in the algorithm input increases, the number of operations grows linearly.

F. Discussion

As stated within the study results, seemingly long messages can be hidden in multiple images that are, in other terms, a visual representation of data. As the proposed method is a mask to hide information, the time it takes to build each image is meaningless compared to the time it takes to encrypt information using RSA. As the input data increases in size, the time it will take to hide the information will rise on a linear scale as stated in the algorithm analysis's complexity, raising concerns about how the algorithm performs on massive amounts of data, and this is the result of the images being built pixel by pixel. Modifications are required to process data in batches, increasing performance and making it a feasible solution to hide massive amounts of corporate data in a reasonable amount of time.

Opposed to traditional steganography, where an already existing image is required to hide sensitive information, which is, in most cases, an image with a considerable size, generating an image from specific data reduces the output file size. This finding is a significant advantage when it comes to storing large amounts of data in an image format. Nevertheless, the current method of hiding data raises clear concerns about security. For instance, the set of rules needed to be specified for the algorithm to work can lead to weak patterns due to human intervention in how the data must be stored. Thus, discovering a pattern could be hard and time-consuming; once it is found and recorded, it is simpler to retrieve data from other images that happen to use the same pattern.

The proposed method's current state opens new paths of increasing the security of hidden data through images. An improved version of the method is needed to use it on real-world applications such as hiding massive amounts of sensitive data or replacing QR codes with efficient data storage public patterns and data generated images. This result leads the investigation into further development of public and private custom patterns to hide information within images compared to public and private keys used on RSA cryptography, increasing the method's security and reliability, making it independent of other cryptography systems. Tree Parity Machines are considered a feasible solution to exchange keys between two parties, using neural cryptography [20] instead of traditional asymmetric cryptography. This method of exchanging data securely through the internet can be modified to train TPM to generate unique, strong patterns each time a new image is built, instead of user-defined rules.

IV. Conclusion

The application of this technique has been achieved successfully, as it reached to hide messages, as it first, encodes them in an encrypted text using RSA. In contrast, afterwards, this text will be converted into an array of bytes stored in the RGB layers of one or more computer-generated images in data transmission. The Theory of Probability and the Java recursion technique has been used to determine the number of images sent and bytes that correspond to each image. The results reveal that the re-encryption is independent of carrier images. The number of different ways to organize the pixels and generated images allows the method to be very secure, as previously demonstrated in the section about" Security level of the algorithm based on Theory of Probability." Moreover, over encryptions, combined with the steganography technique, increases the security level of private messages. As future work, we have planned a pseudo-random method, which selects which layer and pixel will contain relevant data.

Acknowledgement

This scientific research funding is provided by Mobility Regulation of the Universidad de las Fuerzas Armadas ESPE, from Sangolquí, Ecuador. The authors are grateful to the

Ecuadorian Corporation for the Development of Research and the Academy (RED CEDIA) for this study's development within the Project Grant GT-Cybersecurity. Furthermore, this research is part of the Research Project results entitled "Detection and mitigation of Social Engineering attacks using Cognitive Security".

REFERENCES

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[2] Y. Liu and J. Zhang, "Large — Capacity LSB information hiding scheme based on two-dimensional code," in *2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2017, pp. 528–532, doi: 10.1109/ICEIEC.2017.8076621.

[3] S. Mukherjee, S. Sinha, S. Chakrabarti, and T. Mukhopadhyay, "A meticulous implementation of RSA Algorithm using MATLAB for image encryption," in *2017 1st {International} {Conference} on {Electronics}, {Materials} {Engineering} and {Nano}-{Technology} ({IEMENTech})*, 2017, pp. 1–6.

[4] J. Nath and A. Nath, "Advanced Steganography Algorithm using encrypted secret message," *Int. J. Adv. Comput. Sci. Appl.*, vol. 2, no. 3, 2011.

[5] K. Muhammad, J. Ahmad, M. Sajjad, and M. Zubair, "Secure image steganography using cryptography and image transposition," *arXiv Prepr. arXiv1510.04413*, 2015.

[6] S. Almuhammadi and A. Al-Shaaby, "A survey on recent approaches combining cryptography and steganography," *Comput. Sci. Inf. Technol. (CS IT)*, 2017.

[7] S. M. A. Asbeh, S. M. Hammoudeh, and A. Hammoudeh, "AES Inspired Hex Symbols Steganography for Anti-Forensic Artifacts on Android Devices," *vol*, vol. 7, pp. 319–327, 2016.

[8] A. Gutub and K. Alaseri, "Hiding shares of counting-based secret sharing via Arabic text steganography for personal usage," *Arab. J. Sci. Eng.*, pp. 1–26, 2019.

[9] K. Patani and D. Rathod, "Advanced 3-Bit LSB Based on Data Hiding Using Steganography," in *Data Science and Intelligent Applications*, Springer, 2020, pp. 383–390.

[10] Y. Xiang, D. Xiao, R. Zhang, J. Liang, and R. Liu, "Cryptanalysis and improvement of a reversible data-hiding scheme in encrypted images by redundant space transfer," *Inf. Sci. (Ny).*, vol. 545, pp. 188–206.

[11] A. A. AL-Shaaby and T. AlKharobi, "Cryptography and Steganography: New Approach," *Trans. Networks Commun.*, vol. 5, no. 6, p. 25, 2017.

[12] B. Karthikeyan, A. Deepak, K. S. Subalakshmi, A. R. MM, and V. Vaithiyanathan, "A combined approach of steganography with LSB encoding technique and DES algorithm," in *2017 Third Intern. Conf. on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, 2017, pp. 85–88.

[13] M. Khan, S. S. Jamal, and U. A. Waqas, "A novel combination of information hiding and confidentiality scheme," *Multimed. Tools Appl.*, vol. 79, no. 41, pp. 30983–31005, 2020.

[14] S. J. Indrabi, N. Saini, and M. Mohan, "Secure data transmission based on combined effect of cryptography and steganography using visible light spectrum," *Int. J. Pure Appl. Math.*, vol. 118, no. 20, pp. 2851–2860, 2018.

[15] P. Chaudhary, "A Novel Image Encryption Method Based on LSB Technique and AES Algorithm," in *Computational Methods and Data Engineering*, Springer, 2020, pp. 539–546.

[16] N. A. Al-Juaid, A. A. Gutub, and E. A. Khan, "Enhancing PC data security via combining RSA cryptography and video based steganography," *J. Inf. Secur. Cybercrimes Res.*, vol. 1, no. 1, 2018.

[17] S. M. Hardi, M. Masitha, M. A. Budiman, and I. Jaya, "Hiding and Data Safety Techniques in Bmp Image with LSB and RPrime RSA Algorithm," in *Journal of Physics: Conference Series*, 2020, vol. 1566, p. 12084.

[18] E. J. Kusuma, C. A. Sari, E. H. Rachmawanto, and others, "A combination of inverted LSB, RSA, and Arnold Transformation to get secure and imperceptible image steganography," *J. ICT Res. Appl.*, vol. 12, no. 2, pp. 103–122, 2018.

[19] R. H. Thayer, S. C. Bailin, and M. Dorfman, *Software requirements engineerings*. IEEE Computer Society Press, 1997.

[20] É. Salguero Dorokhin, W. Fuertes, and E. Lascano, "On the Development of an Optimal Structure of Tree Parity Machine for the Establishment of a Cryptographic Key," *Secur. Commun. Networks*, vol. 2019, 2019.

[21] T. Fritzmann, T. Pöppelmann, and J. Sepulveda, "Analysis of error-correcting codes for lattice-based key exchange," in *Intern. {Conf.} on {Selected} {Areas} in {Cryptography}*, 2018, pp. 369–390.

[22] M. A. Alia, K. A. Maria, M. A. Alsarayreh, E. A. Maria, and S. Almanasra, "An Improved Video Steganography: Using Random Key-Dependent," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, 2019, pp. 234–237, doi: 10.1109/JEEIT.2019.8717368.

[23] A. J. Ahmad, "Text in Image Steganograghy Using LSD Method," Middle East University, 2014.

[24] W. Fuertes, F. Meneses, L. Hidalgo, and J. Torres, "RSA Over-Encryption Implementation for Networking: A Proof of Concept Using Mobile Devices."

[25] E. C. Molina, "The theory of probability and some applications to engineering problems," *Trans. Am. Inst. Electr. Eng.*, vol. 44, pp. 294–301, 1925.

[26] M. S. Abbas, S. S. Mahdi, and S. A. Hussien, "Security Improvement of Cloud Data Using Hybrid Cryptography and Steganography," in *2020 International Conference on Computer Science and Software Engineering (CSASE)*, 2020, pp. 123–127.

[27] W. Fuertes, J. E. L. de Vergara, F. Meneses, and F. al Galán, "A generic model for the management of virtual network environments," in *2010 IEEE Network Operations and Management Symposium-NOMS 2010*, 2010, pp. 813–816.