Hindawi
Security and Communication Networks
Volume 2018, Article ID 4853134, 16 pages
https://doi.org/10.1155/2018/4853134

Research Article

System to Safeguard the Identity of Persons in Photographs through Cryptography and Steganography Techniques Using Chaos

Octavio Flores Siordia,[1] Juan Carlos Estrada Gutiérrez,[1] Carlos Eduardo Padilla Leyferman,[2] Jorge Aguilar Santiago,[2] and Maricela Jiménez Rodríguez[1]

[1] *División de Desarrollo Biotecnológico, Centro Universitario de la Ciénega, Universidad de Guadalajara, Av. Universidad, No. 1115, Col. Lindavista, Ocotlán, Jalisco, Mexico*
[2] *Centro Universitario de la Ciénega, Universidad de Guadalajara, Av. Universidad, No. 1115, Col. Lindavista, Ocotlán, Jalisco, Mexico*

Correspondence should be addressed to Maricela Jiménez Rodríguez; maricela.jimenez@cuci.udg.mx

Received 18 April 2018; Revised 11 August 2018; Accepted 9 September 2018; Published 4 November 2018

Guest Editor: Ahmed G. Radwan

Safeguarding the identity of people in photographs or videos published through social networks or television is of great importance to those who do not wish to be recognized. In this paper, a face detecting and coding system is designed with the goal of solving this problem. Mathematical models to generate chaotic orbits are deployed. One of them applies the diffusion technique to scramble the pixels of each face while another implements the confusion technique to alter the relation between plain text and ciphered text. Afterward, another two orbits are utilized for the steganography technique to modify the least significant bit (LSB) to conceal data that would allow authorized users to decipher the faces. To verify the robustness of the proposed encryption algorithm, different tests are performed with the Lena standard image, such as correlation diagrams, histograms, and entropy. In addition, occlusion, noise, and plain image attacks are performed. The results are compared with those of other works, and the proposed system provided high sensitivity at secret key and a large space for the encryption keys, good speed for ciphering, disorder in the cryptogram, security, data integrity, and robustness against different attacks.

1. Introduction

Facial recognition techniques are widely used in searching for missing people and enable their hasty identification from information stored in databases. For example, the OpenFace tool performs this task by implementing neural networks [1]. A facial-characteristics extraction dictionary was also developed to enhance the identification process [2]. In addition, a technique fusing two types of virtual sampling with mirror faces and symmetrical faces, added to the original one, was proposed to correct the issue with insufficient samples in the training period and enhanced the recognition rate [3]. However, the detection of biometric characteristics is not always deployed for legal activities: ill-disposed users can access data from other users through information and communication technologies (ICT) in order to, subsequently, harm them.

For this reason, it is crucial to implement techniques to safeguard and protect a person's identity, such as a system in which biometric data is encrypted and then concealed in images by implementing steganography techniques [4]. In other studies, color palettes were exchanged to render zones unrecognizable because when regular palettes are employed, there is a better chance for an attacker to decipher them [5]. Furthermore, fingerprints and facial characteristics were embedded in images through watermark methods [6]. A cryptosystem deploying palm-vein and iris biometrics was proposed. This system extracted characteristics to be encrypted with the Blowfish symmetric algorithm [7]. Another study utilized the Eigenvalue algorithm to identify

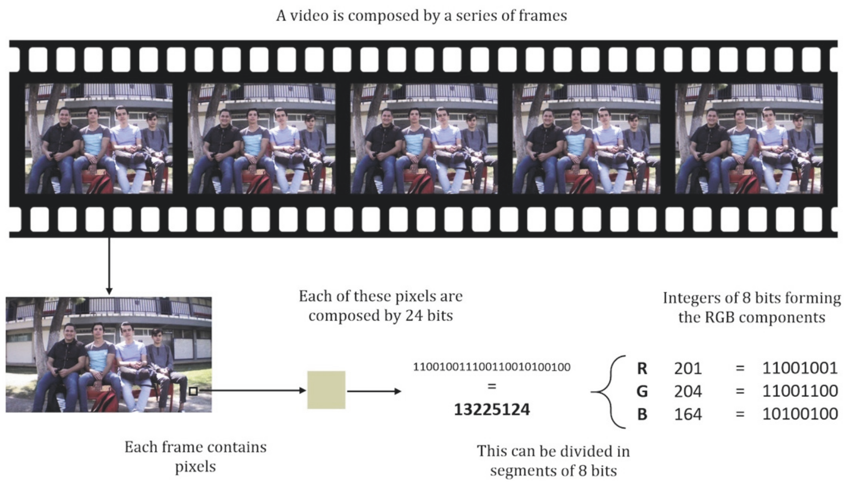

A video is composed by a series of frames
Each of these pixels are composed by 24 bits
Integers of 8 bits forming the RGB components
110010011100110010100100
=
13225124
R 201 = 11001001
G 204 = 11001100
B 164 = 10100100
Each frame contains pixels
This can be divided in segments of 8 bits

Figure 1: Video components.

faces and generate keys from the coding of fingerprints and a header. Later, steganography was employed to conceal the key in the face and, finally, scramble it [8].

For cryptography, chaotic systems can be broadly implemented to encrypt biometric characteristics due to their intrinsic properties, such as ergodicity, mixture, and sensitivity to their parameters and initial conditions [9]. There are studies in which chaotic orbits generated by mathematical models were applied to code data and information [10–13]. It is also possible to employ the technique proposed by Pecora and Carroll in which two chaotic systems can be coupled [14] to encrypt and send data safely once they have been synchronized [15]. Additionally, the logistic map was implemented with a key produced from the characteristics of the iris to cipher [16]. Furthermore, a scheme to encrypt fingerprints to prevent identifying theft was presented. This scheme applies basic encryption techniques, i.e., diffusion and confusion, by employing the hyperchaotic Rössler system [17].

In this research, a system to detect and code faces in videos or photographs is proposed with the purpose of safekeeping identities and preventing them from being easily identified. The proposed technique allows the environment surrounding the faces to remain visible, affording the opportunity for transmission through open television or social networks. Afterward, users can recover the faces as long as they have the keys that were utilized for encryption and steganography. The system detects faces, applies steganography, and implements a new robust and effective encryption system. The novel system employs a key space that can be doubled depending on the security that will be implemented. It also possesses excellent properties against different types of attacks.

The remainder of this paper is organized as follows. In Section 2, the mathematical models implemented in this system are explained, while in Section 3, the face detecting and coding operation system is detailed. Section 4 presents the decryption method. Section 5 contains the results of the tests that were performed. Finally, Section 6 presents the conclusions of this research.

2. Chaotic Mathematical Models

2.1. Logistic Map. This is a simple discrete time system with a nonlinear dynamic that exhibits chaotic behavior and can be expressed by

$$\boldsymbol{xl_{n+1} = \lambda \cdot xl_n \cdot (1 - xl_n)} \tag{1}$$

where the dynamic variable is $xl \in [0,1]$ and λ is the control parameter that, when varied, causes the system to show linear, periodic, or chaotic behavior when $\lambda \in [3.56, 4]$ [18, 19].

2.2. Sine Map. This nonlinear discrete system displays chaotic behavior similar to that of the Logistic map, except that this system involves a sine function with entry values $[\mathbf{0}, \boldsymbol{\pi}]$. This is defined the following equation.

$$\boldsymbol{xs_{n+1} = \sigma \cdot \sin(\pi \cdot xs_n)} \tag{2}$$

The dynamic variable is $\boldsymbol{xs} \in [\mathbf{0}, \mathbf{1}]$ and exhibits chaotic behavior when the control parameter is $\boldsymbol{\sigma} \in [\mathbf{0.87}, \mathbf{1}]$ [18].

3. Face Detecting and Coding System

The proposed system can be implemented to detect and code faces in videos. The videos were divided into the frames that comprise them and each was addressed separately. Afterwards, the pixel values were extracted. These values ranged between 0 and 16777215 in accordance with those corresponding to 24 bits used for red, green, and blue (RGB) combinations and eight bits for each color. This process is presented in Figure 1.

The system operation schematic is shown in Figure 2 and can be explained as follows:

(1) Face detection and extraction from each frame occur first.

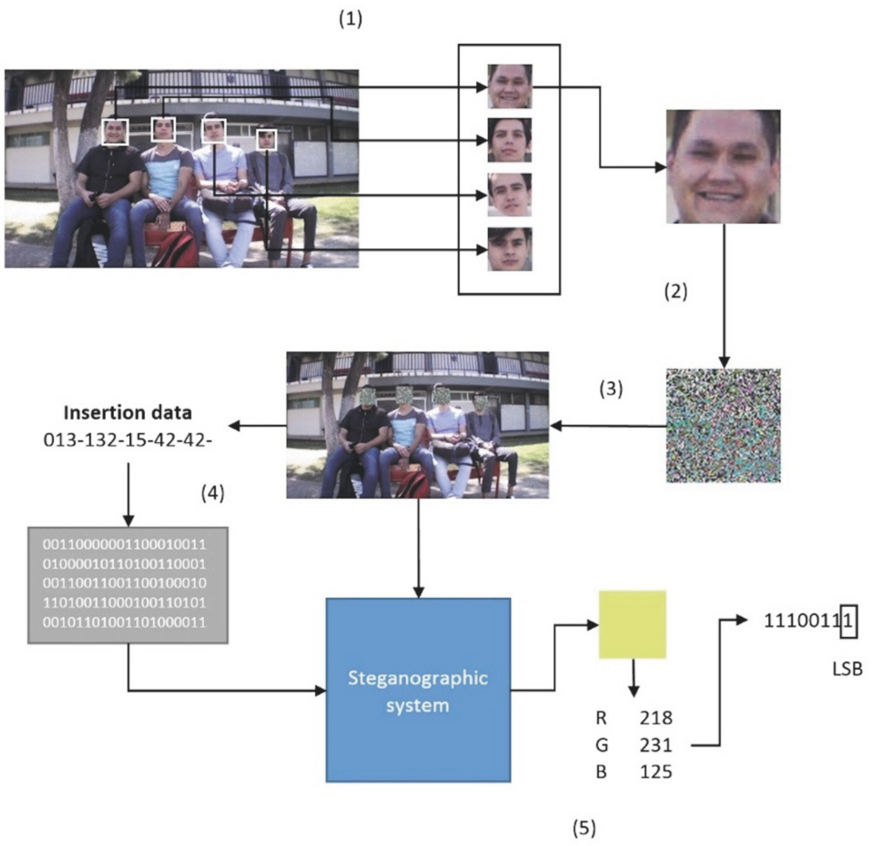


FIGURE 2: Schematic for the general system operation.

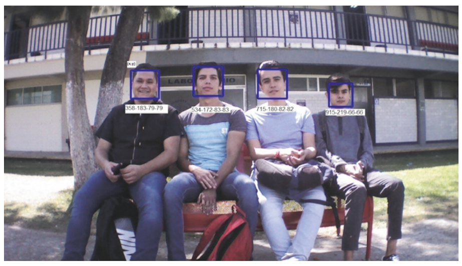


FIGURE 3: Face detection procedure.

(2) Face encryption occurs next.

(3) The cryptogram is then embedded into the original frame.

(4) A message is prepared with the necessary information for decoding.

(5) Steganography that modifies the least significant bit (LSB) is implemented to conceal the data for face decoding.

3.1. Face Detection and Extraction Phase. Face detection was performed with the OpenCV Library, which traces specific objects such as frontal faces, eyes, and full bodies. Once the face has been located, the selection data were generated. Selection data contains the (x, y) pixel coordinates from the selection's initial point, in addition to face width and height. This procedure is illustrated in Figure 3.

Later, the ImageIO Library was employed to cut out the face according to the selection data.

3.2. Encoding Phase. This algorithm requires two orbits for its proper performance, and these orbits can be generated with any chaotic model desired. In this paper, logistic map equation (1) was implemented to apply confusion to the face pixels, while sine map equation (2) was deployed to apply diffusion to the pixels.

Nomenclature Handled for Encryption

OF: The original face selected from the image or frame, in the case of processing a video.

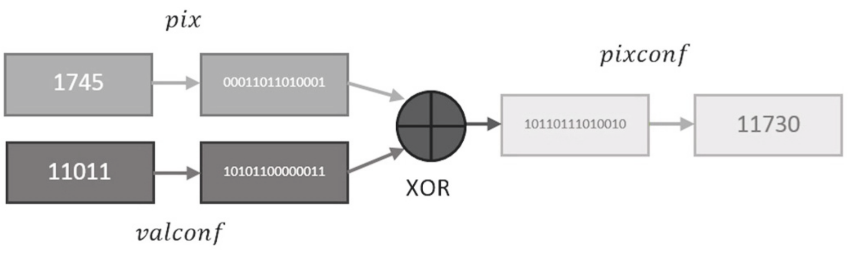


FIGURE 4: Process for applying confusion to the face pixels.

Cryptogram: Codified face with the proposed algorithm.

Ciphering Keys Handled for Cryptogram Generation

n_seg: Number of segments into which the original face (*OF*) will be divided.

λ, xl_0: Parameter and initial condition employed to resolve logistic map discrete system, (1), in its chaotic regimen.

σ, xs_0: Parameter and initial condition deployed to resolve sine map, (2), when the latter generates chaos.

In the following section, the phases implemented to codify the original face (*OF*) are explained.

3.2.1. Confusion Phase. In this phase, the integer values of each pixel were changed by others generated from a chaotic orbit. The steps used in the procedure are as follows:

(1) Calculate the number of pixels inside *OF* by multiplying the width and height of the image. The result is stored under the variable *size*:

$$size = width * height. \tag{3}$$

(2) Retain in the array named *pix* of length *size* the values of all pixels from *OF* consecutively. These values range from 0 to 16777215 according to their color intensity:

$$pix = [53256, 123467, 23454, \ldots, 16654213]. \tag{4}$$

(3) Resolve logistic map equation (1) by implementing the values of the ciphering keys (λ and xl_0) to generate a number between 0 and 1, and the result is assigned to the variable *log*.

(4) Store in *valconf* the resulting value by rounding off the product of multiplying *log* by 16777215:

$$valconf = round\left(log * 16777215\right). \tag{5}$$

(5) Store in the array *pixconf* the integer obtained by the operation *XOR* between a value of *pix* and *valconf*:

$$pixconf\,[i] = pix \oplus valconf. \tag{6}$$

This process is illustrated in Figure 4.

(6) Repeat steps (3) through (5) for each element from the *pix* array. Upon completion, a *pixconf* with *size* quantity of values is generated.

3.2.2. Diffusion Phase. In this phase, the *pixconf* array generated in Section 3.2.1 is fragmented into segments. Diffusion is applied to each segment, scrambling the order of their values. Once all of the parts are processed, they are reassembled to create the cryptogram.

(1) Calculate the number of *spix* pixels from the *pixconf* array corresponding to each segment and divide *size* (total number of elements) by *n_seg*:

$$spix = \frac{size}{n\_seg}. \tag{7}$$

(2) Select the corresponding *spix* values from the current *pixconf* segment.

(3) Calculate a value between 0 and 1 and resolve the system in (2) with its corresponding keys σ and xs_0. The result is stored in *val_s*.

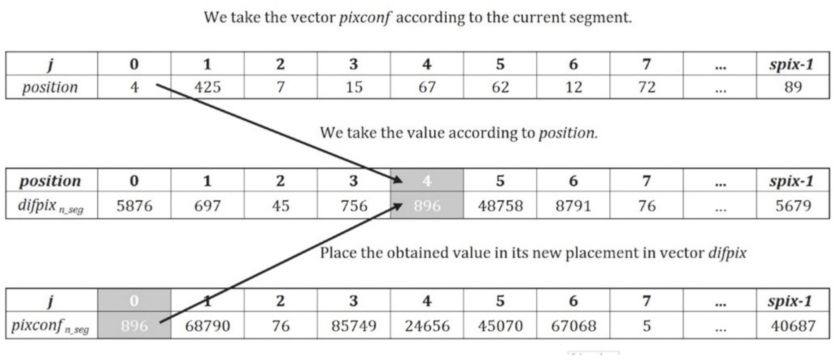
(4) Round off the product of multiplying *val_s* by *spix* to generate an integer between 0 and *spix* − 1. The result is stored in the variable *position*:

$$position = round\left(val\_s * \left(spix - 1\right)\right). \tag{8}$$

(5) Place the value of the current segment from *pixconf* into the *position* location of the same segment number from the *difpix* array. This process is presented in Figure 5. If it is occupied, it is important to verify whether a value has already been generated in position *difpix*. If it is occupied, then a vacated location is searched for by revising the previous placements in *difpix*. If the beginning of the array has been reached and a position is not available. The search continues from the last position until an available location to store the value is found.

(6) Rearrange all values from the current segments in the *difpix* array and repeat steps (3) to (5).

(7) Repeat steps (2) to (6) for each segment until all segments have been scrambled. Afterward, assemble the *cyptogram* (*CRYPTOF*) and arrange the segments of *difpix* generated in step (6) consecutively. This process is shown in Figure 6.



We take the vector *pixconf* according to the current segment.

j	0	1	2	3	4	5	6	7	...	*spix-1*
position	4	425	7	15	67	62	12	72	...	89

We take the value according to *position*.

position	0	1	2	3	4	5	6	7	...	*spix-1*
$difpix_{n\_seg}$	5876	697	45	756	896	48758	8791	76	...	5679

Place the obtained value in its new placement in vector *difpix*

j	0	1	2	3	4	5	6	7	...	*spix-1*
$pixconf_{n\_seg}$	896	68790	76	85749	24656	45070	67068	5	...	40687

FIGURE 5: Scramble and store segment values from $pixconf$ to $difpix$.

3.3. Reincorporation of the Cryptogram in the Photograph. To insert the coded faces, a function called the ImageIO Library was implemented. This function takes the original image, selection data, and the cryptograms generated as entries for its task. Figure 7 depicts this process.

3.4. Message Preparation. Once the cryptograms have been embedded in the image, users would require information on the coordinates and size of each face in order to recover them. Therefore, in this section, the method to generate and conceal a message with this data is explained. The required information is hidden inside the photograph (container image) to send the information unnoticed by a third party by implementing a steganography technique. Thus, only authorized users can decrypt the faces through the use of the correct keys.

The message is composed of two parts: size and data. This is subdivided into several fields according to the total number of detected faces in the image. It has two sections when a single face is detected in the image, three for two detected faces, four for three, and so on. This is shown in Figure 8.

The size conveys a value that indicates the quantity of characters present in the data portion of the message. For example, in Figure 8, the size has a value of 013, meaning that there are thirteen characters in the data portion. This is shown in Figure 9.

The data is stored as explained in detail in Table 1.

In Figure 10, an image containing four faces is shown. Therefore, the number of fields in the data portion increases accordingly.

In Table 2, the information stored in each field according to the selection data from Figure 10 is exhibited.

3.5. Steganography Technique. In this system, a steganography technique proposed by Maricela et al. [26] was implemented.

Nomenclature

Container: The photograph with the encrypted face. The message with the data to decode these will be embedded within the container.

M: Message generated in Section 3.4.

TABLE 1: Information stored in the data portion.

Character	Description
132	Position in x
15	Position in y
42	Face height
42	Face width
-	Data divisor

Steganogram: The *Container* with the hidden message.

The pixels that form the *Container* were divided into their three RGB subpixels of 8 bits each. The three combined convey all of the colors and their different intensities, ranging from 00000000 00000000 00000000, representing black, to 11111111 11111111 11111111, resulting in pure white. These properties were used to apply steganography that deploys the LSB scheme to modify the LSB of each subpixel. The method used chaotic mathematical models to generate two orbits. The first selects the pixel-to-modify and the second selects the RGB subpixel. This procedure is explained briefly below:

(1) Convert M into its respective chain of bits, as shown in Table 3.

(2) Implement the ciphering keys to generate chaotic orbits. These will be deployed to select a subpixel of some RGB color [26]:

$$subp = 10011001. \tag{9}$$

(3) Take a bit from M and compare it with LSB of *subp*. In the case of their being the same, the bit is left unmodified; otherwise, it is replaced. This is presented in Table 4.

(4) Repeat steps (2) and (3) until all the bits of M are embedded in *Container*.

4. Decryption System

To decode the face and return the photograph to its original state, the following phases are required:

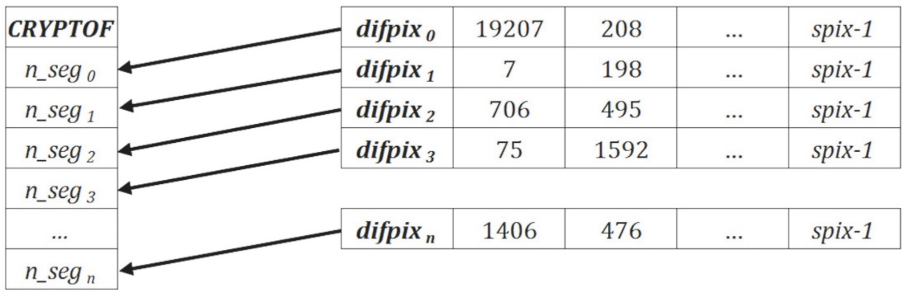

CRYPTOF
n_seg 0
n_seg 1
n_seg 2
n_seg 3
...
n_seg n
difpix 0 | 19207 | 208 | ... | spix-1
difpix 1 | 7 | 198 | ... | spix-1
difpix 2 | 706 | 495 | ... | spix-1
difpix 3 | 75 | 1592 | ... | spix-1
difpix n | 1406 | 476 | ... | spix-1

FIGURE 6: Rearranging the $difpix$ segments for the cryptogram.

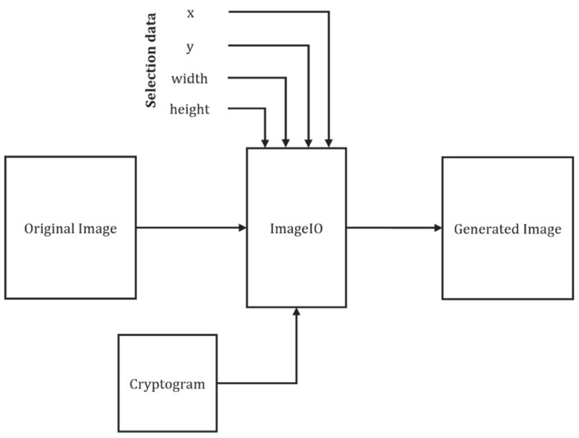
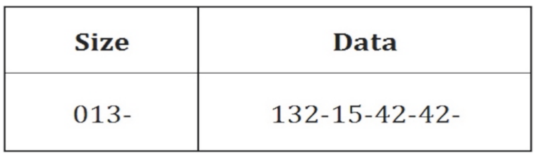
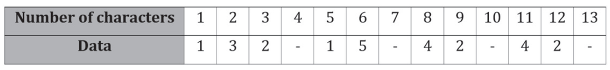


FIGURE 7: Process to reincorporate the faces.



Size	Data
013-	132-15-42-42-

FIGURE 8: Size and data for message sections.



Number of characters	1	2	3	4	5	6	7	8	9	10	11	12	13
Data	1	3	2	-	1	5	-	4	2	-	4	2	-

FIGURE 9: Data on the coordinates and size of the face.

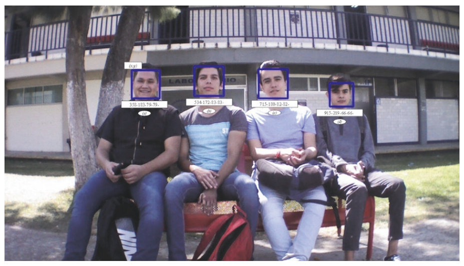


FIGURE 10: Selection data for the image.

4.1. Recovering the Message. Chaotic orbits were generated using the ciphering keys implemented in step (2) of Section 3.5; afterward, the RGB subpixel was selected. Finally, each LSB was taken until M was recovered.

4.2. Extracting Cryptograms. Based on the information about the coordinates, i.e., the height and width of the faces obtained from M in Section 4.1, the cryptograms of the photographs were extracted.

4.3. Decrypting the Cryptograms. For decoding, it is necessary to possess *cryptograms*, the number of segments *n_seg*, and the parameters and initial conditions of each chaotic map $(\lambda, xl_0, \sigma, xs_0)$ used as ciphering keys. Subsequently, the next steps are as follows:

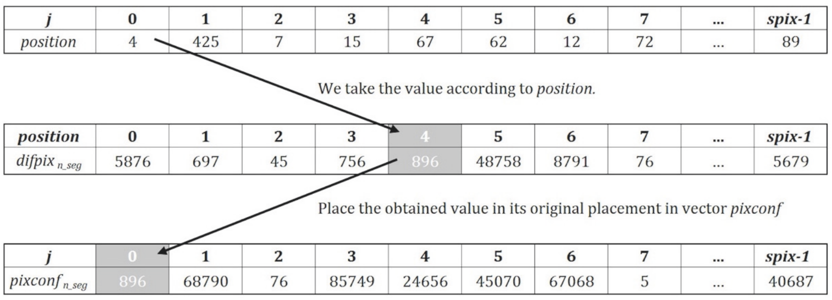
4.3.1. Removing Diffusion. First, the *cryptogram* is segmented. Diffusion is removed by rearranging the values of each of the segments in their original position. The procedure is explained below:

(1) Store the pixels from the *cryptogram* in the $difpix$ vector and the array length *size*.

(2) Define the number of pixels per segment *spix* of the $difpix$ arrays (Step (1) from Section 3.2.2).

Table 2: Data to be stored in the message according to the coordinates of each of the four faces present in Figure 10.

Size	**Data Field**			
056-	358-183-79-79-	534-172-83-83-	715-180-82-82-	915-219-66-66-



j | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | ... | spix-1
position | 4 | 425 | 7 | 15 | 67 | 62 | 12 | 72 | ... | 89
We take the value according to position.
position | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | ... | spix-1
difpix n_seg | 5876 | 697 | 45 | 756 | 896 | 48758 | 8791 | 76 | ... | 5679
Place the obtained value in its original placement in vector pixconf
j | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | ... | spix-1
pixconf n_seg | 896 | 68790 | 76 | 85749 | 24656 | 45070 | 67068 | 5 | ... | 40687

Figure 11: Relocating pixels back to their original position in the current segment.

Table 3: Message conversion into bits.

Message
013-132-15-42-42-
Message in its respective bits
00110000 00110001 00110011 00101101 00110001 00110011 00110010 00101101 00110001 00110101 00101101 00110100 00110010 00101101 00110100 00110010 00101101

Table 4: Modification of the least significant bit (LSB).

***M* Bit**	***Subp***	**Result**
1	1001100$\underline{1}$	1001100$\underline{1}$ (remains same)
0	1001100$\underline{1}$	1001100$\underline{0}$ (replaced)

(3) Select the corresponding *spix* values from the $difpix$ array for the current segment.

(4) Repeat steps (3) and (4) from Section 3.2.2 to generate *position*.

(5) Take the value placed in *position* of the current $difpix$ segment and place it in the next available location of $pixconf$. This is exhibited in Figure 11.

(6) Repeat steps (4) and (5) until all pixels from the current segment have been returned to their original placement. If the value of one *position* in the $difpix$ array has already been relocated, another value is taken from a previous location that has not been moved. In the case of reaching the beginning of the array without finding one, the search continues from the end of the array until one is found. The obtained value is placed in the next empty location of the$pixconf$ array.

(7) Repeat steps (3) to (6) with each segment and place them at the end of *pixconf* until all pixels have been returned to their original location.

4.3.2. Removing Confusion

(1) Repeat steps (3) and (4) of Section 3.2.1 to generate *valconf*.

(2) Apply the *XOR* operation between a value from the *pixconf* and *valconf* arrays to recover the original *OF* pixel value:

$$pix = pixconf\,[i] \oplus valconf. \tag{10}$$

This process is depicted in Figure 12.

(3) Repeat steps (1) and (2) until all of the original values of *pix* have been recovered. The original image *OF* is recovered at the end of this process.

4.4. Face Reincorporation. Faces were reinserted in their original location using Section 3.3 of the face detecting and coding system.

5. Results

In this section, tests and analyses were performed to measure the sturdiness of the proposed encryption technique.

Figure 13(a) displays the original Lena image deployed to test the face detecting and coding process; the resulting image is presented in Figure 13(b). Afterward, the decryption and recovery system were implemented. The recovered image is displayed in Figure 13(c).

Other tests were also performed as shown in Figure 14. Four faces were detected in Figure 14(a), coded as shown in Figure 14(b), and decrypted as shown in Figure 14(c).

5.1. Statistical Analysis. In this section, the correlation diagrams and histograms obtained from the images displayed in Table 5 are exhibited. These images were obtained from Figures 14(a)–14(c).

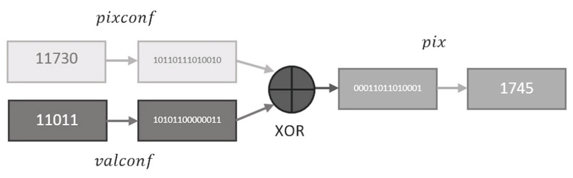
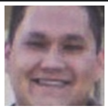
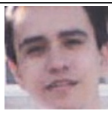
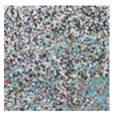
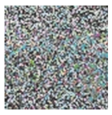


Figure 12: Remove confusion.

Table 5: Original faces, cryptograms, and decrypted faces.

Description	(a)	(b)	(c)	(d)
Original faces				
Cryptograms				
Decrypted faces				

Column (a): original face (a) subtracted by face detection from Figure 14(a); cryptogram generated by proposed algorithm of original (a) from Table 5; and recovered data by proposed algorithm of original (a) from Table 5.
Column (b): original face (b) subtracted by face detection from Figure 14(a); cryptogram generated by proposed algorithm of original (b) from Table 5; and recovered data by proposed algorithm of original (b) from Table 5.
Column (c): original face (c) subtracted by face detection from Figure 14(a); cryptogram generated by proposed algorithm of original (c) from Table 5; and recovered data by proposed algorithm of original (c) from Table 5.
Column (d): original face (d) subtracted by face detection from Figure 14(a); cryptogram generated by proposed algorithm of original (d) from Table 5; and recovered data by proposed algorithm of original (d) from Table 5.

To measure the differences existing between the original faces *vs*. the cryptograms presented in Table 5, the correlation diagrams shown in Figures 15(a)–15(d) were constructed. These R coefficients were equal to -0.02656, -0.03343, -0.02797, and 0.10136, respectively. It can be appreciated that these results are very close to 0; therefore, the correlation between the original images and their cryptograms was practically nonexistent. This indicates that the confusion and diffusion techniques implemented in the system provided sturdiness and complexity, rendering it difficult for an attacker to recover the information by searching for relationships among the original faces, ciphering keys, and cryptograms.

Additionally, the diagrams displayed in Figures 16(a)–16(d) compare the original faces and the deciphered ones in Table 5 to verify the data integrity through the recovery process. The four correlation coefficients of $R = 1$ can be observed. This indicates that the original images were identical to the recovered ones.

The histogram in Figure 17 was obtained from the original face (a) in Table 5 to represent graphically the distribution and frequency of the pixels. The vertical axis displays the number of times the same pixel value repeats across the image, while the horizontal access presents the value corresponding to the color.

In Figure 18, the histogram corresponding to the cryptogram (a) in Table 5 is exhibited.

Comparing the histograms in Figures 17 and 18, it can be observed that there is more homogeneity in the frequencies displayed in Figure 18. This confirms that the system was robust against attacks focused on finding relationships between the original image and the resulting cryptogram to detect the keys used for codification.

The standard Lena image in Figure 13(a) was utilized to perform the histograms shown in Figures 19(a)–19(c) according to RGB. Also, it can be observed in Figures 19(d)–19(f) that the histograms of the image encrypted in its entirety with the proposed system displayed a greater uniformity. Therefore, it would be more difficult for an attacker to determine the relationship between the plain image and the cryptogram.

5.2. Security Analysis

5.2.1. Sensitivity of the Key. To test the sensitivity of the encryption keys, the Lena image was coded with key1 (λ=3.9329, xl_0=0.34908, σ=1.0, xs_0=0.5762, n_seg=50), key2 (λ= 3.9329, xl_0=0.34907, σ=1.0, xs_0=0.5762, n_seg=50), and key3 (λ= 3.9329, xl_0=0.34907, σ=1.0, xs_0=0.5761, n_seg=50). Subsequently, the correlation was carried out between the

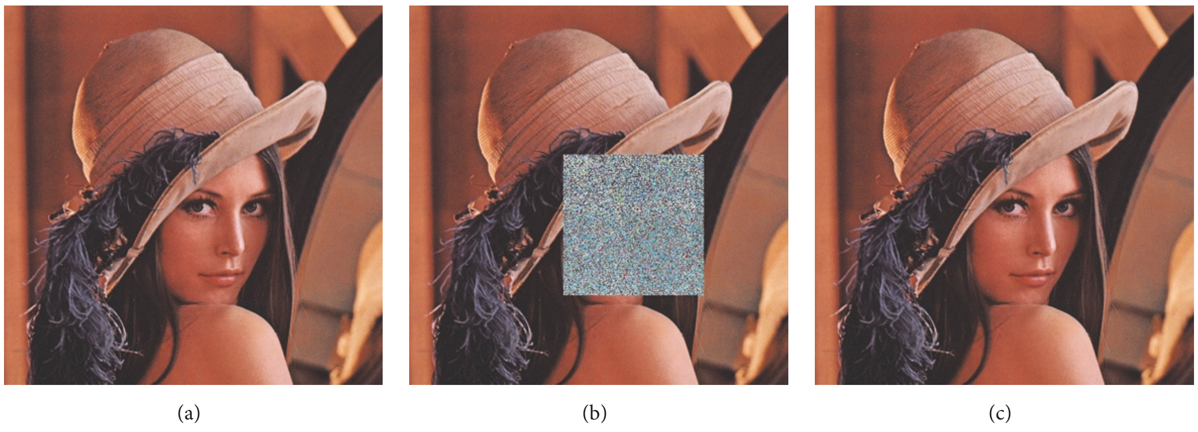


Figure 13: Encryption process: (a) original image, (b) encrypted face, and (c) recovered face. (a) Original Lena test image contained in this article; (b) Lena test image with embedded cryptogram from proposed algorithm; and (c) Lena test image recovered with proposed algorithm.

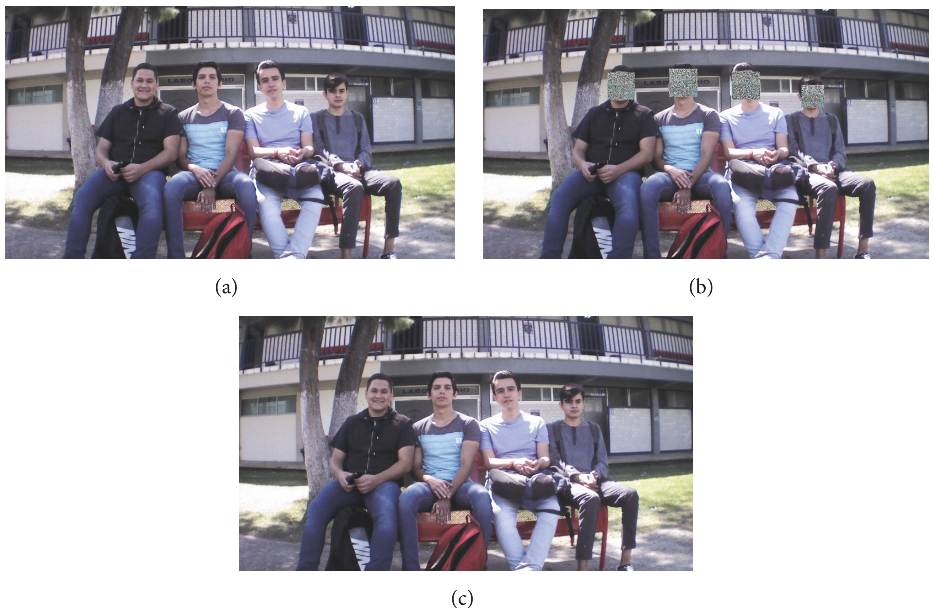


Figure 14: Encryption process: (a) original photograph, (b) image with coded faces, and (c) image with decoded faces. (a) Original photograph used for testing proposed algorithm in this paper; (b) photograph used for testing with embedded cryptograms from proposed algorithm; and (c) Photograph recovered with proposed algorithm.

cryptograms obtained with these keys that all varied by one single digit. Table 6 presents the values of the coefficients obtained and those generated by Murillo et al. [20].

5.2.2. Entropy. To measure the complexity of the encrypted data, an analysis of the entropy of the cryptogram generated from the Lena image was conducted and compared with those mentioned in other investigations. As shown in Table 7, it can be seen that all are close to 8. Therefore, the implemented encryption was robust because it presented a great degree of disorder.

5.2.3. Attacks. To verify the robustness of the system to an attack of the plain image, Figure 20(a) was used. This image was composed of all of its pixels in black to code it with the algorithm and formed in the cryptogram depicted in Figure 20(b). Subsequently, only the confusion technique of the encryption algorithm was implemented in order to cipher the Lena image and resulted in the cryptogram of

Table 6: Comparison on the sensitivity of the key.

Key	Components	Correlation [20]	Proposed
Key1 vs. Key2	R	-0.1281	0.02891
	G	0.0683	0.00129
	B	0.0529	0.01869
Key2 vs. Key3	R	-0.0361	0.00159
	G	0.0856	0.00713
	B	-0.0155	0.00412

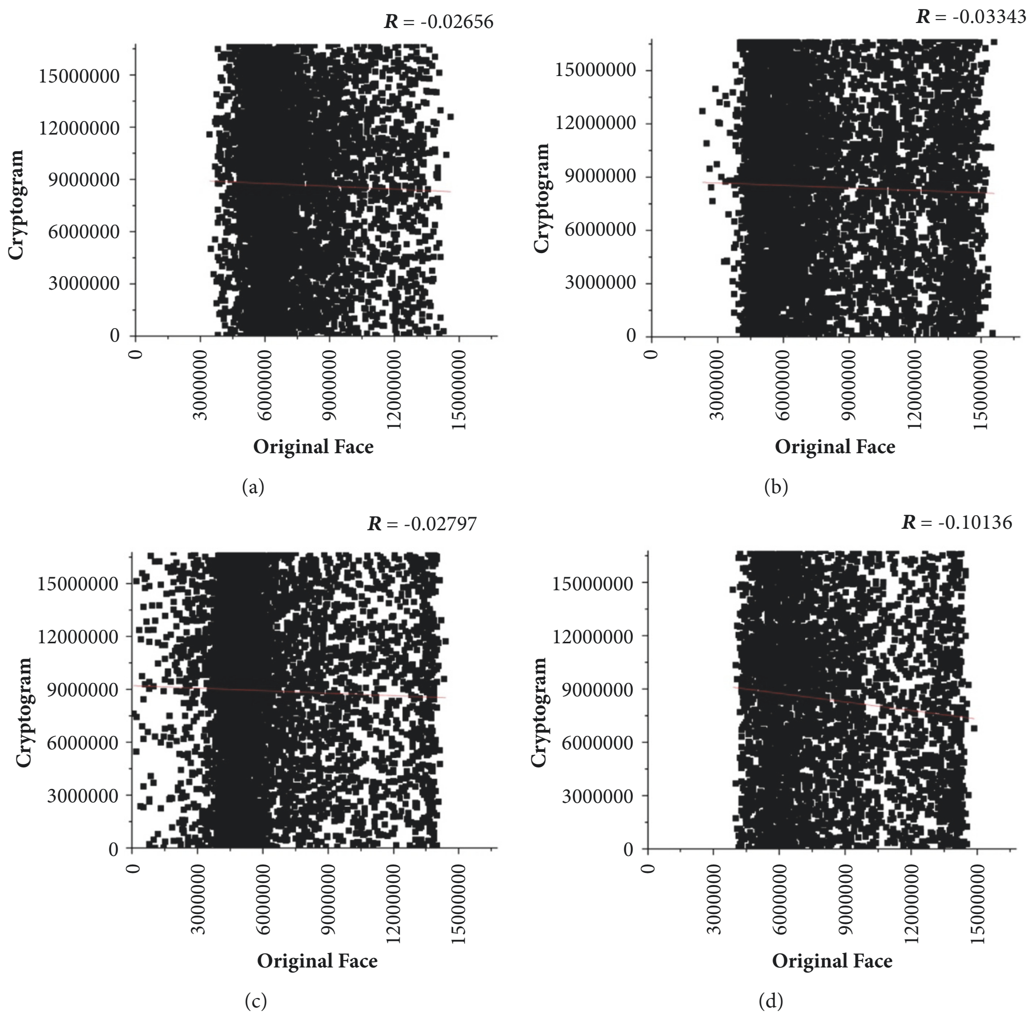


Figure 15: Correlation diagrams for original faces *vs*. cryptograms present in Table 5. (a) Correlation diagram between original (a) from Table 5 and its respective cryptogram. (b) Correlation diagram between original (b) from Table 5 and its respective cryptogram. (c) Correlation diagram between original (c) from Table 5 and its respective cryptogram. (d) Correlation diagram between original (d) from Table 5 and its respective cryptogram.

Table 7: Entropy analysis.

Entropy	Method
7.9855	Proposed
7.9560	[21]
7.9970	[20]
7.9969	[22]

Figure 20(c). Finally, Figure 20(b) was employed as a key to apply the inverse of the confusion technique in an attempt to recover the Lena image and resulted in the image shown in Figure 20(d). Thus, the proposed system was robust against this type of attack.

When a cryptogram is transmitted through a data network, it is possible for it to lose data. For this reason, it is necessary for the proposed system to retrieve the rest of the information. Therefore, it must resist an occlusion attack. In Figure 21(a), the cryptogram is shown with a data loss of 6.25%, and in Figure 21(b), the resulting recovered image is displayed. Figure 21(c) corresponds to the cryptogram with a data loss of 12.5%. When this was deciphered, Figure 21(d) was obtained. Figure 21(e) is the cryptogram with 25% less information and, when deciphered, Figure 21(f) was obtained. Finally, 50% of the data in the cryptogram of Figure 21(g) was eliminated and when it was decrypted, Figure 21(h) was obtained. As can be seen, the Lena image could be deciphered by recovering the remainder of the information even with losses in the cryptogram. In conclusion, the system was resistant to an occlusion attack.

To verify the error between the cryptogram and the original image, the mean square error (MSE) analysis technique was employed. The greater it is, the greater the robustness of the system is. On the other hand, when it approaches 0, the images are very similar. In Table 8, the result of the present system is compared with others.

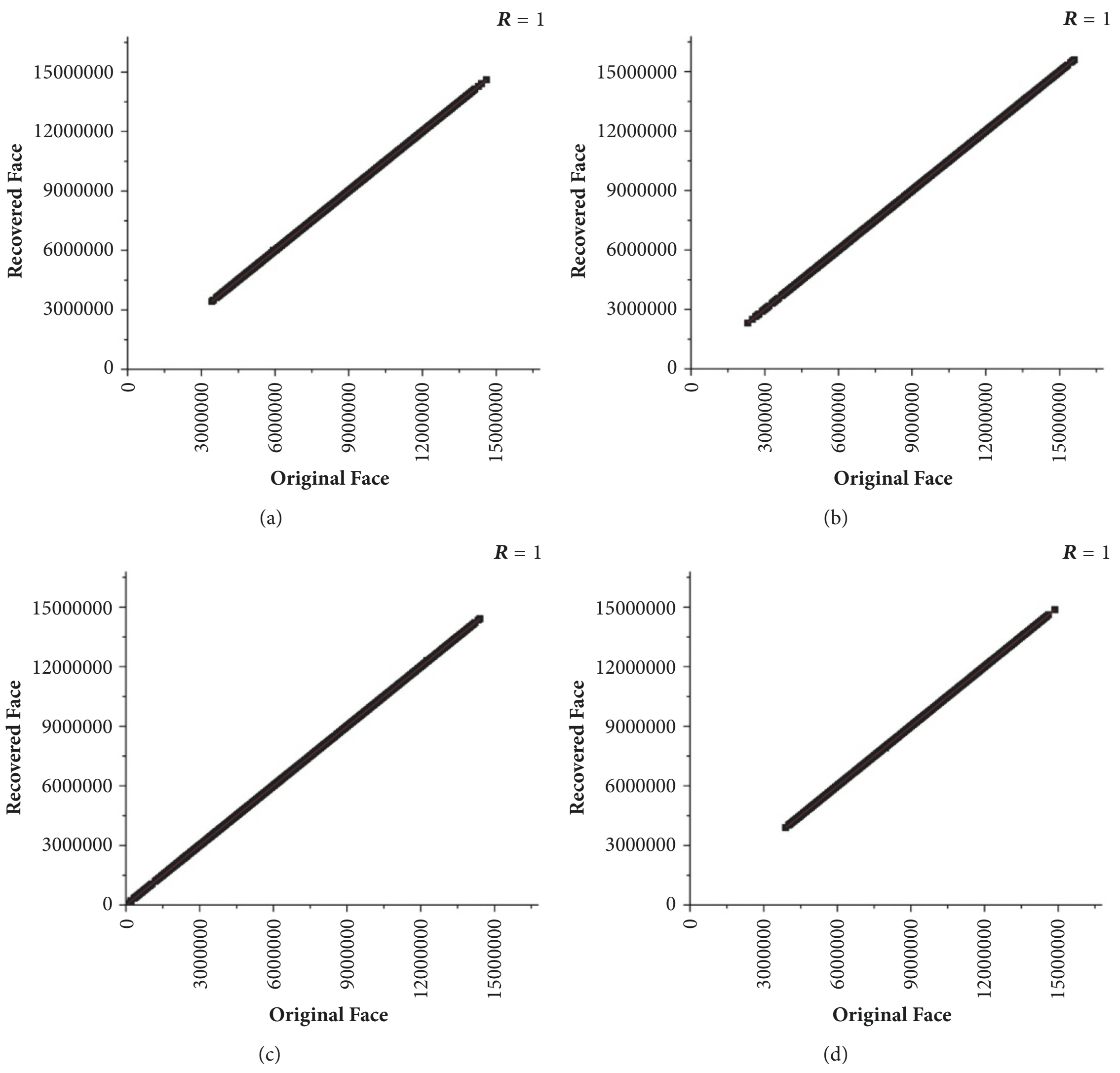


Figure 16: Correlation diagrams for original faces *vs*. recovered faces. (a) Correlation diagram between original (a) from Table 5 and its respective recovered data. (b) Correlation diagram between original (b) from Table 5 and its respective recovered data. (c) Correlation diagram between original (c) from Table 5 and its respective recovered data. (d) Correlation diagram between original (d) from Table 5 and its respective recovered data.

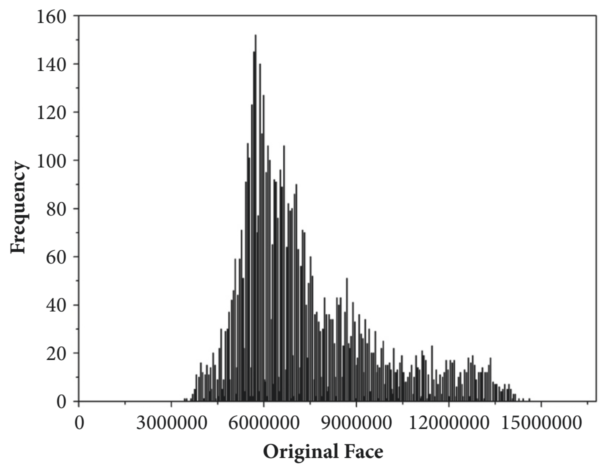


Figure 17: Histogram of the original face (a) shown in Table 5. Histogram measuring the frequency of pixels of original (a) from Table 5.

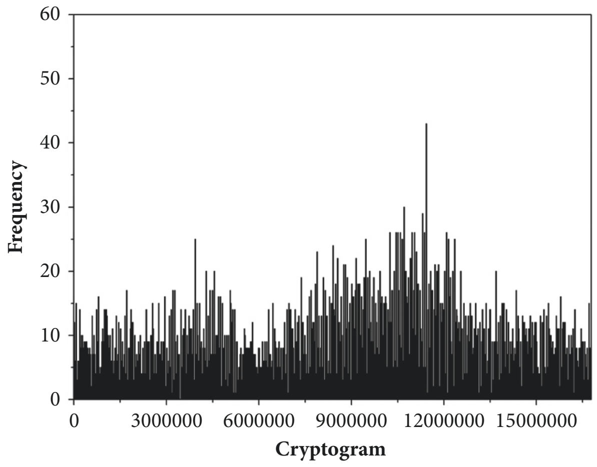


Figure 18: Histogram of cryptogram (a) shown in Table 5. Histogram measuring the frequency of pixels of cryptogram (a) from Table 5.

Table 8: MSE analysis.

MSE	Proposed	[17]	[23]
Original vs. encrypted	9413.4804	12324.3893	10351

Table 9: Comparison of MSE according to the occlusion attack.

Occlusion	Proposed	MSE [17]
6.25%	295.6035	-
12.5%	1221.2244	1844.9842
25%	2432.6524	3581.8797
50%	4860.0209	7101.4179

Table 10: Analysis of the noise attack.

Variance	Proposed	[17]	[22]
0	0	0	-
0.02	2180.5538	5205.101	-
0.1	2940.1990	8129.3254	5631.4354
0.15	3725.8998	8796.4693	-
0.2	4578.1242	9344.5681	-
0.25	5417.8045	9655.1941	-

The MSE was also calculated by comparing Figures 21(b), 21(d), 21(f), and 21(h) with the original. The results are illustrated in Table 9.

An attack was made on the cryptogram of the Lena image where Gaussian noise was incorporated with a standard deviation (SD) ranging from 0 to 0.25. This was later decoded and shown in Figures 22(a)–22(f).

Table 10 exhibits the value of the MSE parameters from Figures 22(a)–22(f) with Gaussian noise with respect to the original image.

5.2.4. Speed Analysis. The encryption-decryption process was implemented in the Java NetBeans ver. 8.2 platform on a laptop with i5-6300HQ CPU at 2.30 GHz, 8.0 RAM, Windows 10 x64 bit OS, Intel® HD Graphics 530 and NVIDIA GeForce GTX 950M Graphics Cards, and ADATA SU800 256 GB solid-state hard drive.

The algorithm proposed in this work quickly encodes and also has the quality of being robust. In Table 11, a comparison of this system with others found in the literature is presented. The latter allows verification of the time of encryption and decryption for each.

5.2.5. Key Space. The encryption system uses four keys (λ, xl_0, σ, and xs_0) that implement a decimal precision of 10^{-15}; therefore, $10^{-15}x10^{-15}x10^{-15}x10^{-15}=10^{-60}$. These are duplicated for each segment indicated in the fifth key (*n_seg*). In addition, when this technique is implemented with the scheme proposed by Maricela et al. [26], another six keys are used, where three correspond to the parameters and another three to the initial conditions of the chaotic system implemented. Therefore, $10^{-15}x10^{-15}x10^{-15}x10^{-15}x10^{-15}x10^{-15}=10^{-90}$. In Table 12, the comparison is shown regarding the encryption key of this system with others.

6. Conclusions

When performing the tests, several issues with the OpenCV Library were encountered. On certain occasions, the library mistakenly identified zones of the image as faces and caused them to be encrypted by the algorithm. Therefore, it was necessary for the photographs to contain frontal faces of good quality. Nonetheless, the system performed face detection, encryption, and decryption in photographs or videos by deploying chaotic cryptography and steganography.

In addition, any chaotic mathematical model can be implemented because this system only requires two orbits for encryption and two additional ones for steganography. It requires three orbits in the case of working with videos, with the latter increasing the number of keys. Further, diffusion was applied by segmenting the image and scrambling each of these segments, and security was independently increased

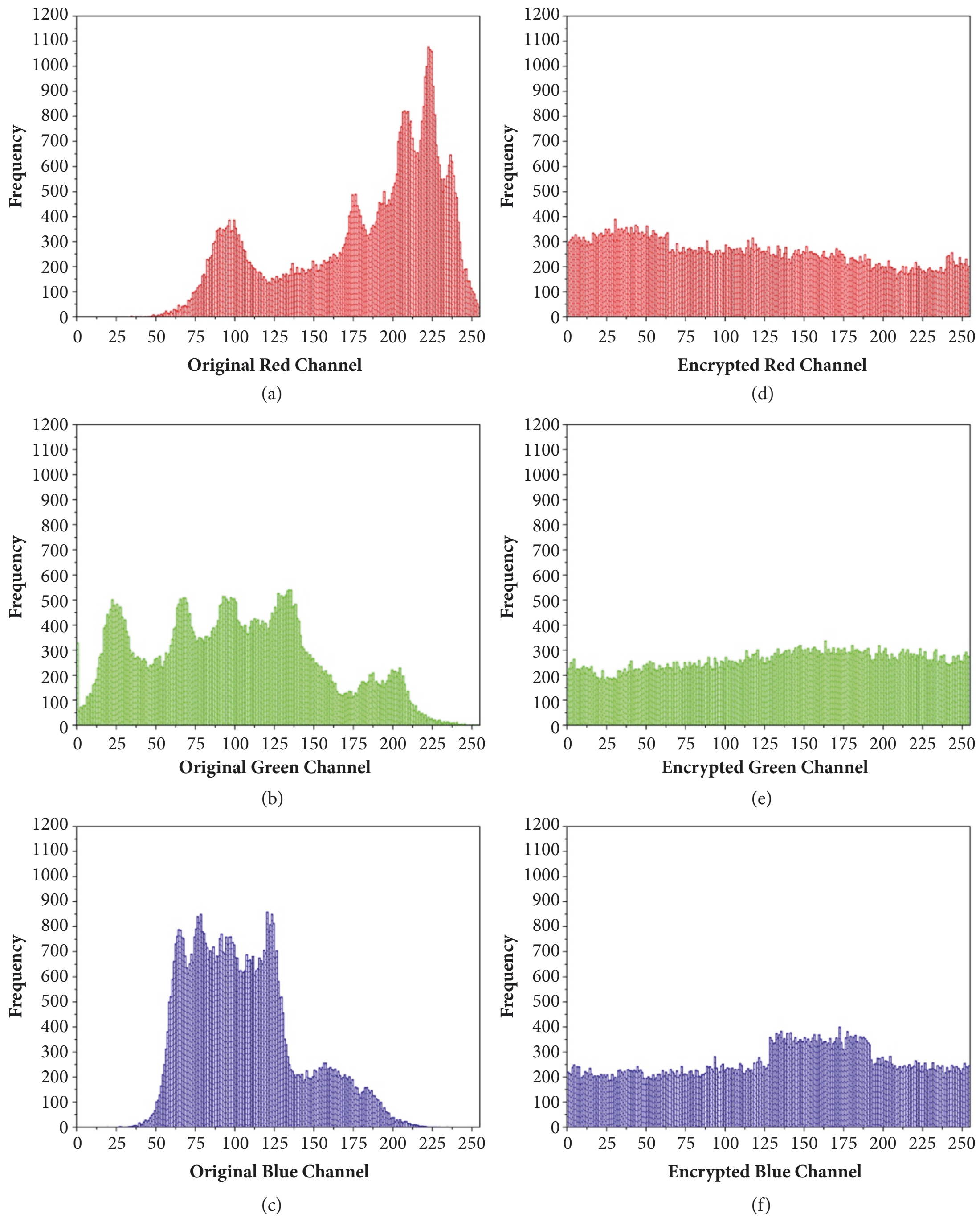


Figure 19: Histogram analysis of encryption process: (a), (b), (c) Lena histograms according to color, with (d), (e), (f) generated RGB cryptograms. (a) Red histogram of the image of Lena in Figure 13(a). (b) Green histogram of the image of Lena in Figure 13(a). (c) Blue histogram of the image of Lena in Figure 13(a). (d) Red histogram of the cryptogram of Lena. (e) Green histogram of the cryptogram of Lena. (f) Blue histogram of the cryptogram of Lena.

Table 11: Encryption speed comparison.

Time (s)	**Proposed 256x256**	**Proposed 512x512**	**[21]**	**[20] 256x256**	**[20] 512x512**	**[17] 355x390**
Cipher	0.062	0.334	1,67342	0.0657	0.2432	0.2340
Decryption	0.046	0.156	1,30321	Similarly to cypher	Similarly to cypher	0.2105

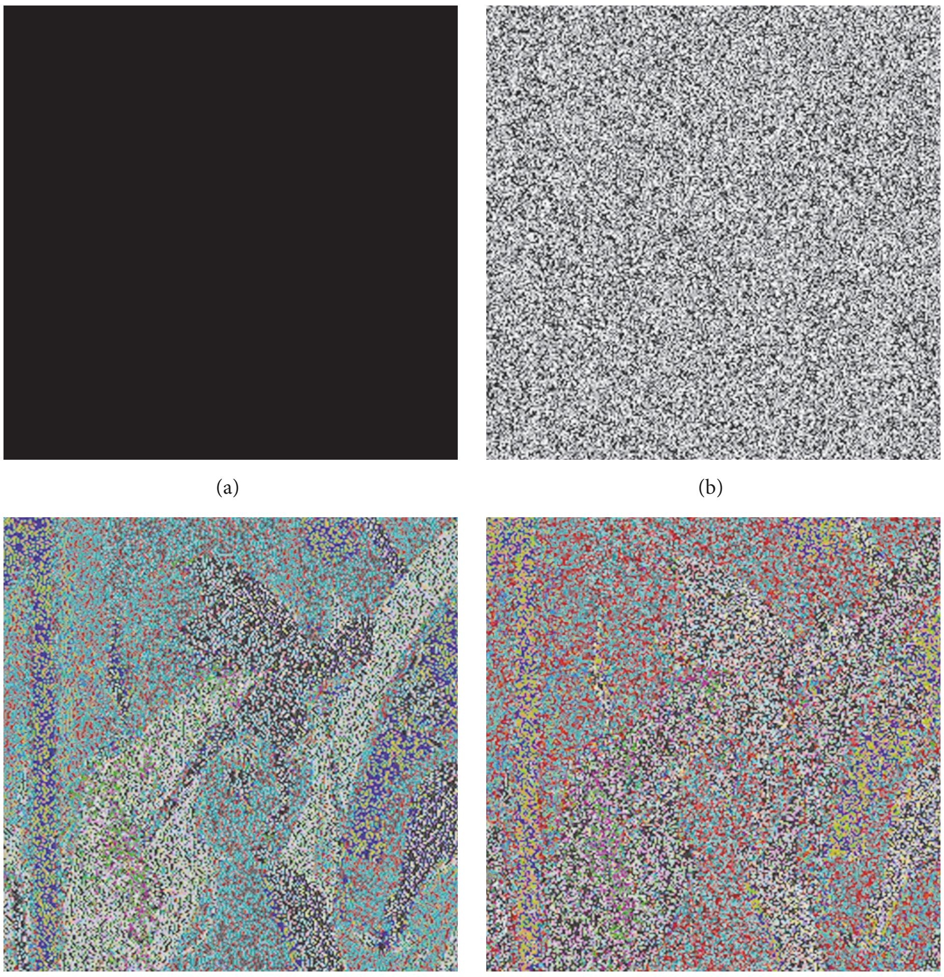


Figure 20: Attack of the plain image: (a) black image, (b) black image cryptogram, (c) Lena cryptogram using confusion, and (d) recovery of the decryption. (a) Black image used for attack. (b) Black image cryptogram generated with the proposed confusion technique. (c) Lena cryptogram generated with the proposed confusion technique. (d) Image deciphered using the confusion technique using Figure 20(b).

Table 12: Key space comparison.

Analysis	**Proposed**	**[21]**	**[20]**	**[17]**	**[24]**	**[25]**
Key space	10^{-60} cypher x *n_seg* 10^{-90} steganography	10^{98}	2^{128}	10^{150}	128 bits	128 bits

by requiring an attacker to know the pixel number in each segment and the key used for decryption.

Another remarkable feature of the system is that it does not encrypt the photograph entirely and allows publication in media without showing the individuals appearing in them. In addition, the system prevents facial recognition by humans or systems and maintains integrity by recovering 100% of the information in the decryption process. On the other hand, when comparing the system with others, it was observed that this system was highly sensitive to encryption keys and the entropy was very close to 8, although other references provided a slight improvement in the latter. It was also robust in terms of occlusion and noise attacks and also permits increasing the number of keys depending on the segments in which it is encrypted. Finally, it demonstrated excellent speed for encryption and decryption.

Data Availability

The algorithms, figures, and pixel data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

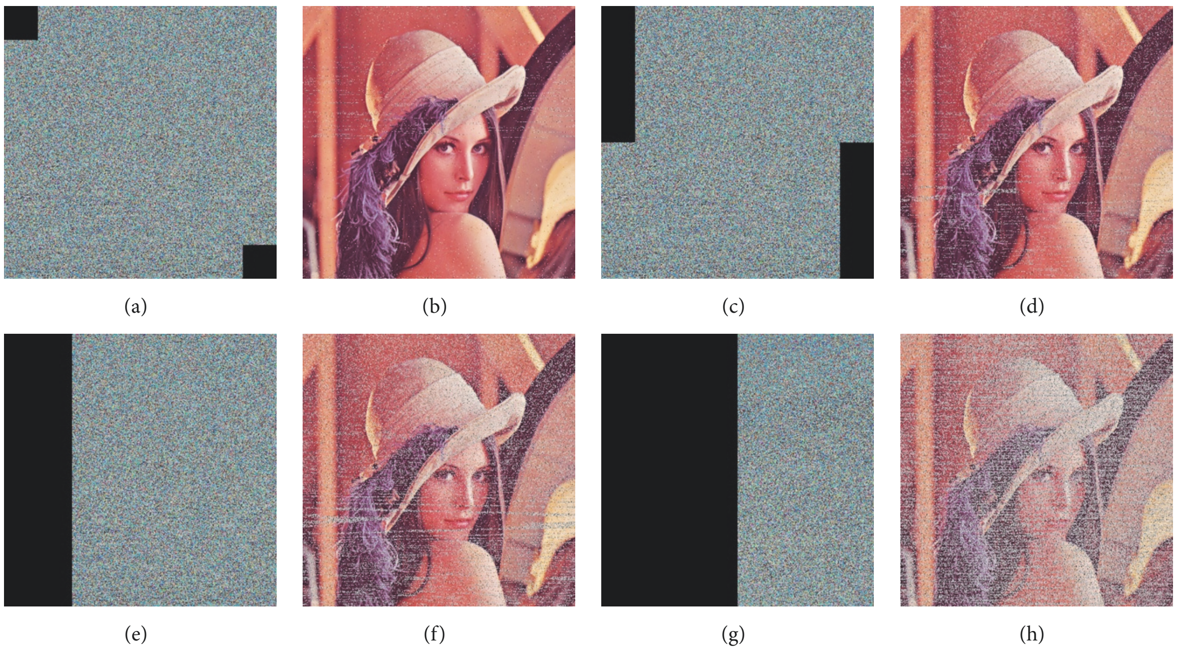


Figure 21: Occlusion attack. (a)–(f) Gaussian noise with variance ranging from 0 to 0.25. (a) Cryptogram with 6.25% loss of information. (b) Image recovered of the cryptogram of Figure 21(a). (c) Cryptogram with 12.5% loss of information. (d) Image recovered of the cryptogram of Figure 21(c). (e) Cryptogram with 25% loss of information. (f) Image recovered of the cryptogram of Figure 21(e). (g) Cryptogram with 50% loss of information. (h) Image recovered from the cryptogram of Figure 21(g).

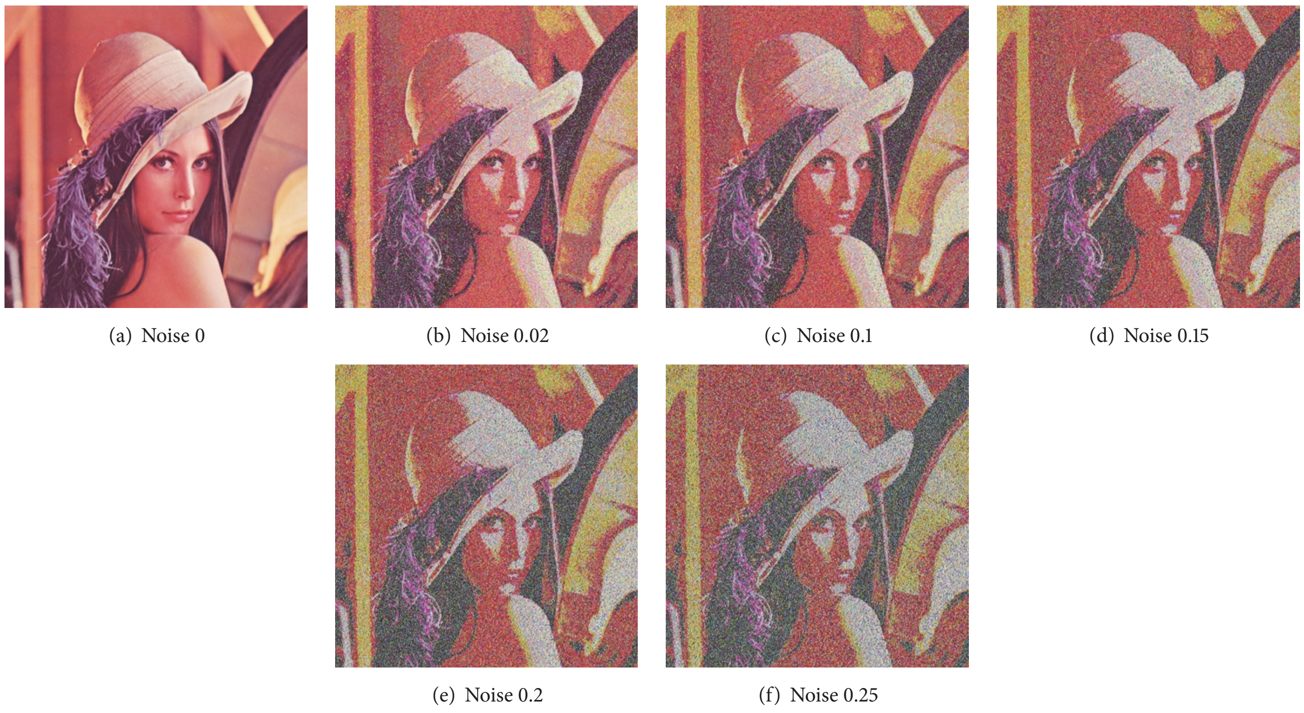


Figure 22: Noise attack. (a)–(f) Gaussian noise with variance ranging from 0 to 0.25. (a) Image of Lena with 0% noise. (b) Image of Lena with 0.02% noise. (c) Image of Lena with 0.1% noise. (d) Image of Lena with 0.15% noise. (e) Image of Lena with 0.2% noise. (f) Image of Lena with 0.25% noise.

Acknowledgments

This work is supported by CONACYT in Project 743187 and 887565.

References

[1] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "OpenFace: A general-propuse face recognition library with mobile applications," 2016, https://cmusatyalab.github.io/openface/.

[2] P Buyssens and M Revenu, "Visible and infrared face identification via sparse representation," *ISRN Machine Vision*, vol. 2013, Article ID 579126, 10 pages, 2013.

[3] M. Qiu, Z. Jian, J. Ynag, and L. Ye, "Fusing two kinds of virtual samples for small sample face recognition," *Mathematical Problems in Engineering*, vol. 2015, Article ID 280318, p. 10, 2015.

[4] N. Agrawal and M. Savvides, "Biometric data hiding: A 3 factor authentication approach to verify identity with a single image using steganography, encryption and matching," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009*, pp. 85–92, Miami, Fla, USA, June 2009.

[5] S. Ciftci, A. O. Akyuz, and T. Ebrahimi, "A Reliable and Reversible Image Privacy Protection Based on False Colors," *IEEE Transactions on Multimedia*, vol. 20, no. 1, pp. 68–81, 2018.

[6] A. K. Jain and U. Uludag, "Hiding biometric data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 11, pp. 1494–1498, 2003.

[7] B. K. Bala and J. L. Joanna, "Multi modal biometrics using cryptographic algorithm," *European Journal of Academic Essays*, vol. 1, no. 1, pp. 6–10, 2014.

[8] B. Shanthini and S. Swamynathan, "Multimodal biometric-based secured authentication system using steganography," *Journal of Computer Science*, vol. 8, no. 7, pp. 1012–1021, 2012.

[9] C. Fu, B. Lin, Y. Miao, X. Liu, and J. Chen, "A novel chaos-based bit-level permutation scheme for digital image encryption," *Optics Communications*, vol. 284, no. 23, pp. 5415–5423, 2011.

[10] A. Jolfaei and A. Mirghadri, "Image encryption using chaos and block cipher," *Computer and Information Science*, vol. 4, no. 1, pp. 172–185, 2011.

[11] M. Amin, O. S. Faragallah, and A. A. Abd El-Latif, "A chaotic block cipher algorithm for image cryptosystems," *Communications in Nonlinear Science and Numerical Simulation*, vol. 15, no. 11, pp. 3484–3497, 2010.

[12] J. Maricela, F. Octavio, and G. M. Guadalupe, "Sistema para codificar información implementando varias órbitas caóticas," *Ingeniería, Investigación y Tecnología*, vol. 16, no. 3, pp. 335–343, 2015.

[13] H. E.-D. H. Ahmed, H. M. Kalash, and O. S. Farag Allah, "An Efficient Chaos-Based Feedback Stream Cipher (ECBFSC) for image encryption and decryption," *Informatica (Ljubljana)*, vol. 31, no. 1, pp. 121–129, 2007.

[14] L. M. Pecora and T. L. Carroll, "Synchronization in chaotic systems," *Physical Review Letters*, vol. 64, no. 8, pp. 821–824, 1990.

[15] M. Jiménez-Rodríguez, M. G. González-Novoa, J. C. Estrada-Gutiérrez, C. Acosta-Lúa, and O. Flores-Siordia, "Secure point-to-point communication using chaos," *DYNA (Colombia)*, vol. 83, no. 197, pp. 181–187, 2016.

[16] A. S. Alghamdi, H. Ullah, M. Mahmud, and M. K. Khan, "Bio-chaotic stream cipher-based iris image encryption," in *Proceedings of the 7th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, EUC 2009*, pp. 739–744, Vancouver, Canada, August 2009.

[17] F. Abundiz-Pérez, C. Cruz-Hernández, M. A. Murillo-Escobar, R. M. López-Gutiérrez, and A. Arellano-Delgado, "A fingerprint image encryption scheme based on hyperchaotic Rössler map," *Mathematical Problems in Engineering*, vol. 2016, Article ID 2670494, 15 pages, 2016.

[18] Y. Zhou, L. Bao, and C. L. P. Chen, "A new 1D chaotic system for image encryption," *Signal Processing*, vol. 97, pp. 172–182, 2014.

[19] S. Al-Maadeed, A. Al-Ali, and T. Abdalla, "A new chaos-based image-encryption and compression algorithm," *Journal of Electrical and Computer Engineering*, vol. 2012, Article ID 179693, 11 pages, 2012.

[20] M. A. Murillo-Escobar, C. Cruz-Hernández, F. Abundiz-Pérez, R. M. López-Gutiérrez, and O. R. Acosta Del Campo, "A RGB image encryption algorithm based on total plain image characteristics and chaos," *Signal Processing*, vol. 109, pp. 119–131, 2015.

[21] Ü. Çavuşoğlu, S. Kaçar, I. Pehlivan, and A. Zengin, "Secure image encryption algorithm design using a novel chaos based S-Box," *Chaos, Solitons & Fractals*, vol. 95, pp. 92–101, 2017.

[22] Z. Parvin, H. Seyedarabi, and M. Shamsi, "A new secure and sensitive image encryption scheme based on new substitution with chaotic function," *Multimedia Tools and Applications*, vol. 75, no. 17, pp. 10631–10648, 2016.

[23] M. Khan and T. Shah, "An efficient chaotic image encryption scheme," *Neural Computing and Applications*, vol. 26, no. 5, pp. 1137–1148, 2015.

[24] M. A. Murillo-Escobar, C. Cruz-Hernández, F. Abundiz-Pérez, and R. M. López-Gutiérrez, "A robust embedded biometric authentication system based on fingerprint and chaotic encryption," *Expert Systems with Applications*, vol. 42, no. 21, pp. 8198–8211, 2015.

[25] L. Teng, X. Wang, and J. Meng, "A chaotic color image encryption using integrated bit-level permutation," *Multimedia Tools and Applications*, pp. 1–14, 2017.

[26] M. Jiménez Rodríguez, C. E. Padilla Leyferman, J. C. Estrada Gutiérrez, M. G. González Novoa, H. Gómez Rodríguez, and O. Flores Siordia, "Steganography applied in the origin claim of pictures captured by drones based on chaos," *Ingeniería e Investigación*, vol. 38, no. 2, pp. 61–69, 2018.