ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/360196362

An Enhanced ElGamal Cryptosystem for Image Encryption and Decryption

Conference Paper · March 2022

DOI: 10.1109/CSASE51777.2022.9759643

CITATIONS
0

READS
59

4 authors, including:

Haval Ismael Hussein
University of Zakho
8 PUBLICATIONS **55** CITATIONS
SEE PROFILE

Ramadhan J. Mstafa
University of Zakho
37 PUBLICATIONS **640** CITATIONS
SEE PROFILE

Younis Younis
University of Zakho
3 PUBLICATIONS **22** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Video Steganography View project

Project Steganography View project

All content following this page was uploaded by Haval Ismael Hussein on 06 July 2022.

The user has requested enhancement of the downloaded file.

An Enhanced ElGamal Cryptosystem for Image Encryption and Decryption

Haval I. Hussein
Dept of Computer Science,
University of Zakho
Zakho, Kurdistan Region, Iraq
barwary.haval@gmail.com

Ramadhan J. Mstafa [1,2]
[1]*Dept of Computer Science,*
University of Zakho
[2]*Dept of Computer Science,*
Nawroz University
Duhok, Kurdistan Region, Iraq
ramadhan.mstafa@uoz.edu.krd

Abdulhakeem O. Mohammed
Duhok Polytechnic University
Duhok, Kurdistan Region, Iraq
abdulhakeem.mohammed@dpu.edu.krd

Younis M. Younis
Dept of Computer Science,
University of Zakho
Zakho, Kurdistan Region, Iraq
younis.younis@uoz.edu.krd

2022 International Conference on Computer Science and Software Engineering (CSASE) | 978-1-6654-2632-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/CSASE51777.2022.9759643

***Abstract*—ElGamal cryptosystem is one of the well-known public-key algorithms for its ability to generate different ciphertexts for the same plaintext on successive runs. However, this algorithm results in a ciphertext occupying a larger memory space than its plaintext due to its encryption nature. As a result, it is pretty infeasible to use data that require their encrypted form to have the same size, such as image data. To overcome this issue, we propose an enhanced ElGamal cryptosystem that can be used for any given digital data message, including image, text, and video. The proposed approach mainly tests image data, consisting of three stages: key pair generation, image encryption, and image decryption. First, we generate as many random bytes as required for encrypting or decrypting images using the sender or receiver's public key information. Then, we use an XOR operation between each pixel in the image and each randomly generated byte to obtain the encrypted or decrypted image. Experimental results revealed that the proposed approach gives excellent results in various evaluation metrics tested on four different color images.**

Keywords—ElGamal cryptosystem, Asymmetric cryptography, Image encryption, Image decryption, Security

I. INTRODUCTION

As Internet and multimedia technology advances, digital images, with their inherent qualities of intuitiveness and vividness, are becoming increasingly popular to disseminate multimedia information. This raises concerns about the safety of image transmission over communication networks. The specific properties of images, such as block data capacity, high redundancy, and strong correlation between neighboring pixels, make image encryption different from text encryption [1], [2].

Different approaches for ensuring the privacy and security of image data have been suggested in the last few years. Symmetric and asymmetric approaches are two common categories of cryptography distinguished by how keys are distributed. Asymmetric cryptography employs a public key for encryption and a private key for decryption, whereas symmetric cryptography uses the same key [3], [4]. Symmetric cryptography techniques provide the advantages of minimal complexity and rapid speed. On the other hand, symmetric cryptography's key management and distribution is a significant challenge. In other words, the number of keys required to encrypt data using the symmetric cryptography technique grows exponentially with each subsequent usage. As a result, encrypted communication is slowed down and becomes more expensive [5].

This difficulty can be solved by using asymmetric cryptography, which requires the encryption key and the decryption key to be separated and unrelated. There are two sorts of keys in asymmetric cryptography: public and private keys. Unlike the private key, which is accessible exclusively to the intended recipient, the public key is made available [6]. Fig.1 illustrates the general concept of this approach. This figure shows that the receiver keeps the private key while the transmitter receives the public key. Then, the transmitter encrypts the data with the public key. Finally, the ciphertext is transferred to the recipient, who uses the private key to decode it. Since only the recipient has access to the private key in this approach, managing and distributing the keys becomes much more manageable. Hence, this approach has attracted the attention of many researchers to developing image cryptosystem approaches that make use of its concept.

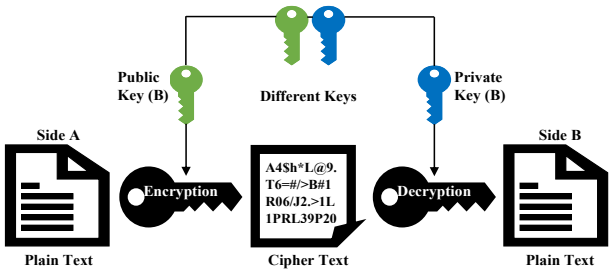

Fig. 1. An illustration of asymmetric cryptography.

Wu *et al*. [7] developed an asymmetric image encryption technique that utilizes a minimal number of keys and key groups in secret information transmission among multiple users. The plain image was initially compressed and then encrypted using an enhanced 4D Cat Map in this technique. After that, the encrypted image was further encrypted using an elliptic curve ElGamal encryption and then globally diffused. They concluded that their proposed approach is more secure than other approaches using various evaluation metrics.

Authorized licensed use limited to: Kent State University Libraries. Downloaded on April 26,2022 at 10:37:40 UTC from IEEE Xplore. Restrictions apply.

Hayat and Azam [8] proposed a hybrid image encryption approach based on a dynamic S-box and pseudo-random numbers (PRN) over an ordered elliptic curve (EC). In this approach, S-boxes were generated using the x-coordinates of the order EC followed by modulo 256 operation. Whereas PRN was generated using the concepts of a generalized Frobenius map and n-norm on order EC. To encrypt a pain image, they initially masked it with the PRN and then confused it with the S-boxes. They claimed from rigorous analysis and comparisons with relevant existing approaches that their proposed approach could generate cryptographically strong S-boxes, PRN with high entropy and optimal resistance against modern image cryptanalysis.

Luo *et al.* [9] developed a new asymmetric image encryption approach based on an elliptic curve ElGamal (EC-ElGamal) cryptography and chaotic theory. In this approach, the SHA-512 hash was initially used to generate the seed values of a chaotic map. Then the plain image was scrambled using crossover permutations defined according to the chaotic index sequence. Afterwards, the scrambled image was further encrypted using EC-ElGamal and then diffused based on a chaos game with DNA sequence. Experimental results indicated that their proposed approach exhibits high security, efficiency, and resistance against a chosen-plaintext attack.

Banik *et al.* [10] proposed an encryption approach for medical images based on a new elliptic curve analogue, ElGamal cryptosystem and Mersenne Twister pseudo-random number generator. They claimed that their approach reduces the encryption time as well as solves the data expansion issue linked to the ElGamal cryptosystem. The simulation, security, and statistical analyses showed that their proposed encryption approach could be applied to multiple medical images.

From the literature, it is clear that most of the previous studies have worked on addressing some of the issues related to the original ElGamal cryptosystem (OEC). The issues that accompany the OEC are summarized as follows. First, encrypting a given message with the OEC results in an encrypted message having a larger size when compared to its original one. This is especially true when the prime number utilized is substantially greater than the maximum Unicode representation of the given message. Second, the OEC is considered computationally expensive and resource-consuming due to the above issue. Lastly, when the same Unicode representation appears multiple times in the given message, the OEC produces the same encrypted Unicode representation multiple times. For instance, if message='eee', then its encrypted message='sss'. This is because the OEC employs this formula $z_i = (y^k \times m_i)\ mod\ p$ to encrypt the entire message, where y^k is constant for all m_i.

Motivated by the facts mentioned above, in this paper, we intend to propose an enhanced ElGamal cryptosystem (EEC) by making some modifications to the encryption and decryption processes of the OEC. These modifications are performed to address the aforementioned issues while achieving the same level of security. Even though the proposed EEC can be applied to any type of digital data, this paper is mainly concerned with encrypting and decrypting image data.

The following are the key contributions of this paper to the field of information security:

- The proposed approach can be used for any given digital data message, including images, text, video, etc. The resultant encrypted message outputs the same size as its original message.
- The proposed approach generates a set of bytes with a size approximately equal to the number of bytes in the given message during the encryption process. This is accomplished by using the public key information from the user to whom the encrypted message is to be sent.
- To prevent the encrypted message from having a larger size than its original message during the encryption process, we propose to perform an XOR operation between each byte of the original message and one byte of a randomly generated set of bytes at a time.

Following is the rest of the paper's structure. Section II describes the ElGamal cryptosystem in its original state. While Section III reports the proposed approach for image encryption and decryption in detail. Then, Section IV includes experimental results as well as a discussion of the results. Finally, Section V concludes the paper and suggests some future remarks.

II. ELGAMAL CRYPTOSYSTEM

This cryptosystem is regarded as one of the most secure non-deterministic schemes since it generates different ciphertexts for the same plaintext on successive runs. It is based on the Diffie-Hellman key exchange protocol, which depends heavily on solving the discrete logarithm problem for its security to be compromised. This problem arises when the public key information (p, α, y) is provided, and subsequently, the private key (x) is revealed, as shown in (1). So far, no efficient method has been developed to solve the discrete logarithm problem with a large prime number that could be feasible for cryptanalysis. As a result, this cryptosystem is considered highly secure for large prime numbers [11], [12].

$$y = \alpha^x\ mod\ p \tag{1}$$

Key pair generation, message encryption, and message decryption are the three main stages of this cryptosystem [13], briefly described in the following subsections.

A. Key Pair Generation

During this stage, the receiver, who later receives an encrypted message, establishes the public key and private key information required for the encryption and decryption processes by following the steps given below:

- Generate p (a large prime number).
- Initialize α (a primitive root of p) so that $1 < \alpha < p - 1$.
- Initialize x (a random integer) so that $1 < x < p - 2$.
- Compute y as in (1).

Once the key pair is generated, the receiver forwards only the public key information (p, α, y) to the sender so that the private key (x) remains secret.

Authorized licensed use limited to: Kent State University Libraries. Downloaded on April 26,2022 at 10:37:40 UTC from IEEE Xplore. Restrictions apply.

B. Message Encryption

Upon receiving the public key information (p, α, y) from the receiver, the sender encrypts a given message according to the steps given below:

- Transform each character in the given message (m) to its corresponding Unicode representation so that $0 \leq m_i \leq p-1$.
- Initialize k (a random integer representing the private key) so that $1 < k < p-2$.
- Compute d as in (2).
- Use the formula given in (3) to encrypt each m_i, where $i = 0, 1, 2, \ldots, L-1$, and L represents the length of m.

$$d = \alpha^k \, mod \, p \tag{2}$$

$$z_i = (y^k \times m_i) \, mod \, p \tag{3}$$

Once the encryption process is completed, the sender forwards the ciphertext information (d, z) to the receiver while keeping his private key confidential. Where, d refers to the sender's public key, and z refers to the encrypted message.

C. Message Decryption

After receiving the ciphertext information (d, z) from the sender, the receiver decrypts the encrypted message according to the steps given below:

- Compute r as in (4).
- Use the formula given in (5) to decrypt each z_i.

$$r = d^{(p-1-x)} \tag{4}$$

$$m_i = (r \times z_i) \, mod \, p \tag{5}$$

III. The Proposed Enhanced ElGamal Cryptosystem

This section provides a detailed description of the proposed EEC for image encryption and decryption. The proposed EEC includes three stages, namely key pair generation, image encryption, and image decryption, as shown in Fig. 2. The first stage of the proposed EEC operates similarly to the OEC, but the encryption and decryption processes operate differently. In these two proposed stages, we intend to generate as many bytes as required for encrypting or decrypting any given image using the public key information provided by the sender or the receiver. Accordingly, both parties can obtain the same shared secret key, which helps them encrypt or decrypt any given image.

Furthermore, we use an XOR operation during the encryption and decryption processes since it is easy to implement and computationally inexpensive. This is achieved by performing an XOR operation between each byte of the randomly generated set of bytes and each pixel in the given image at a time. The following subsections provide an in-depth explanation of the three stages of the proposed approach.

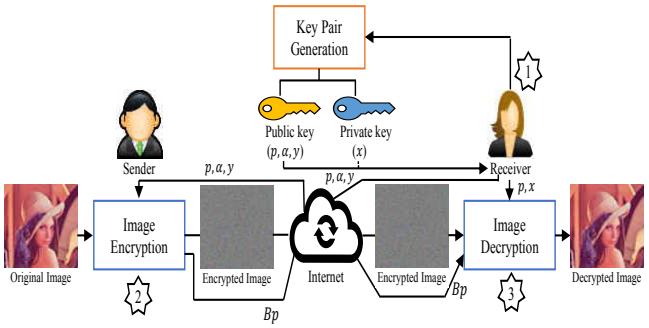

Fig. 2. A block diagram of the proposed Enhanced ElGamal cryptosystem.

A. Key Pair Generation

The proposed EEC follows the same steps as the OEC for generating key pairs (Section II-A). To demonstrate this, an example is provided below.

For simplicity, we assume that p, α, x are randomly selected as 997, 809, and 420, respectively. These values can then be used to compute y, as in (1): $809^{420} \, mod \, 997 = 12$. Where, p, α, and y are the receiver's public key information, and x is the receiver's private key.

As soon as the key pair is generated, the receiver's public key information is sent to the sender while the receiver's private key is kept confidential.

B. Image Encryption

Following the steps given in Algorithm 1, we could encrypt any given image. These steps are divided into three main parts: reading and initializing variables (Step 1 to Step 3), generating the bytes required for encryption (Step 4 to Step 9), and the encryption process (Step 10).

Algorithm 1: Image Encryption

Input:	im, p, α, and y.
Output:	z and Bp.
Step1.	Read im, p, α, and y.
Step2.	Initialize Bp and fsk as follows: $Bp = []$ and $fsk = []$.
Step3.	Initialize z with zeros, so its shape and type are identical to im.
Step4.	Select k (a random integer) such that $1 < k < p-2$.
Step5.	Compute d as in (2).
Step6.	Append d into Bp.
Step7.	Compute sk as in (6).
Step8.	Convert the sk into a set of bytes and then append each byte separately into fsk.
Step9.	If len(fsk) is less than len(im), go to step4.
Step10.	Compute z as in (7) to obtain an encrypted image.

To demonstrate these major parts, an example is provided as follows. Assume that the original image to be encrypted has a resolution of 3×3 as in Fig. 3-I, and the public key information received as in Section III-A ($p = 997$, $\alpha = 809$, and $y = 12$). Subsequently, consider initializing the following variables as $Bp = []$, $fsk = []$, and $z = \text{zeros}(3,3)$. Where Bp, fsk, and z refer to the sender's final public key, the final shared secret key, and the image filled with encrypted pixels, respectively.

$$sk = y^k \, mod \, p \tag{6}$$

$$z_{i,j,c} = im_{i,j,c} \oplus fsk_t \tag{7}$$

Authorized licensed use limited to: Kent State University Libraries. Downloaded on April 26,2022 at 10:37:40 UTC from IEEE Xplore. Restrictions apply.

12	66	23
204	138	76
0	94	51

I)

16	65	252
205	148	190
2	15	75

II)

Fig. 3. Encryption results of the above hypothetical example using the proposed approach: I) an original image and II) an encrypted image of the original one.

To randomly generate a set of bytes with a size roughly equal to the number of pixels in the original image, Step 4 to Step 9 from Algorithm 1 are considered. These steps should generate at least nine bytes since we have nine pixels in our example. The results of these steps are listed below.

- Suppose in step4, $k_1 = 87, k_2 = 578, k_3 = 734, k_4 = 55, k_5 = 376, and\ k_6 = 622$ are randomly selected to be the sender's private keys.
- According to step5, $d_1 = 320, d_2 = 619, d_3 = 122, d_4 = 273, d_5 = 171, d_6 = 918$, where d represents the sender's public keys.
- According to step6, $Bp = [320, 619, 122, 273, 171, 918]$.
- According to step7, $sk_1 = 796, sk_2 = 491, sk_3 = 30, sk_4 = 754, sk_5 = 81, sk_6 = 888$, where sk represents the shared secret key.
- According to step8, $sk_1 = [28,\ 3]$, $sk_2 = [235, 1]$, $sk_3 = [30], sk_4 = [242, 2], sk_5 = [81], sk_6 = [120, 3]$. This step converts each shared secret key into a set of bytes. For instance, the first shared secret key ($sk_1 = 796$) is transformed as follows: $796_d = 0000\ 0011\ 0001\ 1100_b = [28, 3]$.
- According to step8, $fsk = [28, 3, 235, 1, 30, 242, 2, 81, 120, 3]$.

According to step 9, all previous steps (4 to 8) are performed six times. This is to have an approximately equal number of bytes (shared secret bytes) required for implementing the encryption process.

To encrypt the original image given in Fig. 3-I, Step10 from Algorithm 1 is considered. The results of this step are given in Fig. 3-II. It is important to note that the size of the fsk in some cases exceeds the size of the original image, as in our example. To avoid this problem, we simply neglect the extra bytes obtained. At the end of this stage, the sender sends the encrypted image (z) and his final public key (Bp) to the receiver.

C. Image Decryption

Using the steps listed in Algorithm 2, we could decrypt any given image. As in the encryption steps, these steps are as well broken down into three main parts: reading and initializing variables (Step 1 to Step 3), generating the bytes required for encryption (Step 4 to Step 5), and the encryption process (Step 6).

Algorithm 2: Image Decryption

Input: $z, Bp, p,$ and x.

Output: im.

Step 1. Read $z, Bp, p,$ and x.
Step 2. Initialize fsk as follows: $fsk = []$.
Step 3. Initialize im with zeros, so its shape and type are identical to z.
Step 4. Divide Bp into a set of blocks as follows: $Bp_1, Bp_2, \dots, Bp_{n-1}$.
Step 5. For each block Bp_i, do the following:
  1) Compute sk as in (8).
  2) Convert the sk into a set of bytes and then append each byte separately into fsk.
Step 6. Compute $im_{i,j,c}$ as in (9) to obtain the original image.

To demonstrate these major parts, an example is provided as follows. Assume that the encrypted image is decrypted as in Fig. 3-II, and the public key received as in Section III-B ($Bp = [320, 619, 122, 273, 171, 918]$). Subsequently, consider initializing the following variable as $fsk = []$, $im = \text{zeros}\ (3,3)$, $p = 997$, and $x = 420$.

$$sk = Bp_i{}^x\ mod\ p \tag{8}$$

$$im_{i,j,c} = z_{i,j,c} \oplus fsk_t \tag{9}$$

To generate the same set of bytes used during the encryption process, Step 4 to Step 5 from Algorithm 2 are considered. The results of these steps are listed below.

- According to step4, $Bp_1 = 320, Bp_2 = 619, Bp_3 = 122, Bp_4 = 273, Bp_5 = 171, Bp_6 = 918$.
- According to step5-1, $sk_1 = 796, sk_2 = 491, sk_3 = 30, sk_4 = 754, sk_5 = 81, sk_6 = 888$.
- According to step5-2, $sk_1 = [28,\ 3]$, $sk_2 = [235, 1]$, $sk_3 = [30], sk_4 = [242, 2], sk_5 = [81], sk_6 = [120, 3]$.
- According to step5-2, $fsk = [28, 3, 235, 1, 30, 242, 2, 81, 120, 3]$.

To decrypt the encrypted image given in Fig. 3-II, Step 6 from Algorithm 2 is considered. The results of this step are given in Fig. 3-I.

IV. Experimental Results and Discussion

The proposed approach was validated through an extensive set of experiments conducted on four benchmark color images with a size of 512×512×3 (see Fig. 4-I). All reported results were obtained from a computer equipped with Intel(R) Core (TM) i7-8550U CPU 1.99GHz and 12GB of RAM.

To better reflect the performance of the proposed approach, several security tests were considered, including the visibility test, histogram, encryption and decryption speed, entropy, and correlation coefficient. For more details on the description of the aforementioned security tests, the reader is referred to [14], [15]. The results of these tests were based on the main variables used in the proposed approach described in Table 1.

Authorized licensed use limited to: Kent State University Libraries. Downloaded on April 26,2022 at 10:37:40 UTC from IEEE Xplore. Restrictions apply.

TABLE I A DESCRIPTION OF THE VARIABLES USED IN THE PROPOSED APPROACH.

User	Stage	Variable used	Description	No. of bits	Justification
Receiver	Key Pair Generation & Image Decryption	p	A large prime number	1024	For a high-security level
		α	a primitive root of p	1020	
		x	Receiver's private key	1000	
		y	Receiver's public key	<=1024	Because of $mod\ p$ operation
Sender	Image Encryption	$k_1, k_2, \dots, k_n$	Sender's private keys	Each 1000	For a high-security level. Note that the number of private keys to be used depend on the input image
		$d1_1, d1_2, \dots, d_n$	Sender's public keys (Bp)	Each <=1024	Because of $mod\ p$ operation. Note that the number of public keys to be used depend on the input image

Fig. 4 shows the original tested images along with their encrypted and decrypted images. From this figure, it is evident that the encrypted images are noise-like images. Moreover, it can be observed that the decrypted images are identical to the original images. This means that the proposed approach produces unintelligible images to unauthorized parties while restoring the original images without causing any loss to the authorized parties.

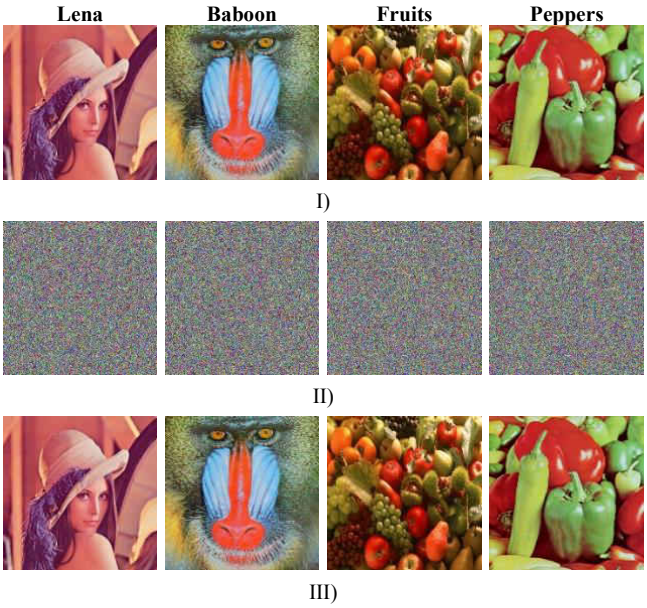

Fig. 4. Visibility test: I) Original images; II) Encrypted images; III) Decrypted images.

Table 2 denotes the entropy value of all tested images before and after the encryption. As can be seen from this table that the entropy value for all cipher images is very close to the maximum value of 8, which means the encrypted images of the proposed method has a random pixel value distribution.

Table 3 shows the correlation coefficient for all tested images before and after the encryption for each color channel in three directions (horizontal, vertical, and diagonal directions). It can be seen from the results that the values of two adjacent pixels are similar to the plain images. However, the values of two adjacent pixels in the encrypted images are different, and the correlation coefficients of the proposed approach are close to 0.

Fig. 5 shows the time taken for encrypting and decrypting each tested image using the proposed approach. It is clear from this figure that the elapsed time for the encrypted images is twice that of the decrypted images. This shows that the proposed approach performs better during the decryption process.

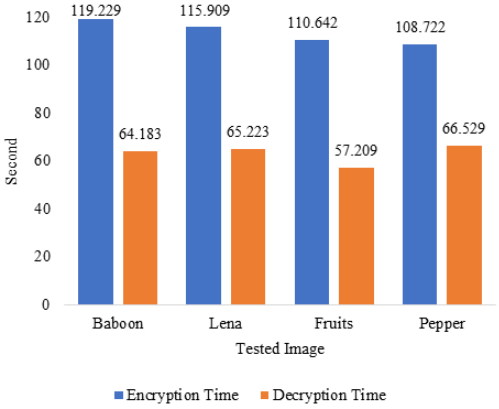

Fig. 5. Time required for encrypting and decrypting each tested image using the proposed approach.

TABLE II ENTROPY RESULTS

Image	Original Image	Encrypted Image
Lena	7.7502	7.9998
Baboon	7.7624	7.9998
Fruits	7.6542	7.9998
Peppers	7.6698	7.9998

TABLE III CORRELATION COEFFICIENT OF THE ORIGINAL AND ENCRYPTED IMAGES FOR EACH CHANNEL

Image	Direction	Original Image			Encrypted Image		
		Red	Green	Blue	Red	Green	Blue
Lena	Horizontal	0.9812	0.9747	0.9484	-0.0739	-0.0434	-0.0413
	Vertical	0.9901	0.9848	0.9624	0.0071	-0.0182	-0.0443
	Diagonal	0.9745	0.9559	0.9269	-0.0473	-0.0298	-0.0618
Baboon	Horizontal	0.9317	0.8876	0.9159	-0.0118	-0.0362	-0.0885
	Vertical	0.8729	0.7983	0.8883	-0.0222	-0.0213	-0.0265
	Diagonal	0.8684	0.7628	0.8502	-0.0170	-0.0805	-0.0490
Fruits	Horizontal	0.9713	0.9627	0.9526	-0.0548	-0.0520	-0.0077
	Vertical	0.9785	0.9756	0.9595	-0.0466	-0.0303	-0.0190
	Diagonal	0.9635	0.9583	0.9363	-0.0078	-0.0145	-0.0176
Peppers	Horizontal	0.9644	0.9844	0.9731	-0.0594	-0.0563	-0.0215
	Vertical	0.9659	0.9837	0.9709	-0.0577	-0.0739	-0.0127
	Diagonal	0.9626	0.9678	0.9514	-0.0252	-0.0643	-0.0625

Authorized licensed use limited to: Kent State University Libraries. Downloaded on April 26,2022 at 10:37:40 UTC from IEEE Xplore. Restrictions apply.

Fig. 6 shows histograms of the original tested images along with their encrypted images. From this figure, it can be noted that the histograms of the encrypted images are totally different from their original images. Furthermore, the histogram distributions of the encrypted images are almost uniform. This indicates that the proposed approach is resistant to statistical attacks.

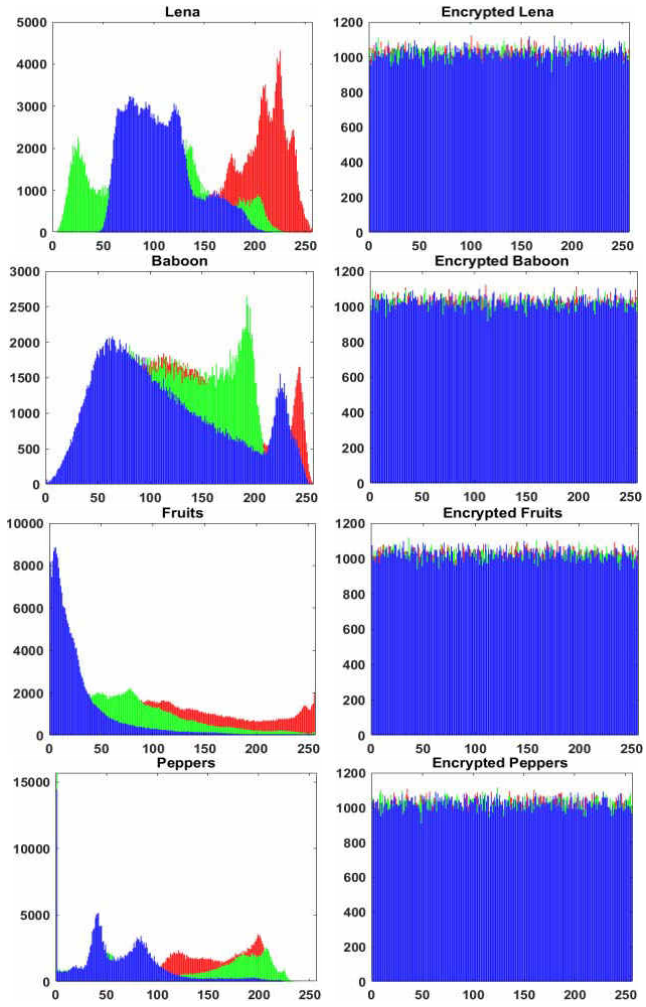

Fig. 6. Histogram of the original images and encrypted images.

Despite our modifications to the OEC, the proposed approach ensures the same level of security. This is because it also relies on solving the discrete logarithm problem, which there is no known efficient method to break when the prime number used is as large as 1024 bits.

V. Conclusions and Future Directions

This paper proposes an efficient image cryptography method called EEC, based on the OEC. The proposed approach has several advantages over the OEC. First, it results in an encrypted image having the same size as the original image. Second, the time required for its encryption and decryption processes is much faster since it uses an XOR operation between each pixel and each byte generated randomly rather than using a multiplication operation with a large number used by the OEC. Third, it can be applied to any digital data, such as text, video, and audio. Ultimately, the randomly generated set of bytes during the encryption and decryption processes are determined by the input image size and the number of bits used for the main variables of the proposed approach. Empirical results on four benchmark color images demonstrated that the proposed approach yields excellent results in terms of visibility test, histogram analysis, entropy test, correlation coefficient test, and encryption and decryption speed. For future work, we aim to use different evaluation metrics to indicate the performance of the proposed approach further. Moreover, we intend to apply it to other digital data, such as video and audio, to see the extent of its feasibility for those data.

References

[1] Y. Luo, L. Cao, S. Qiu, H. Lin, J. Harkin, and J. Liu, "A chaotic map-control-based and the plain image-related cryptosystem," Nonlinear Dynamics, vol. 83, pp. 2293–2310, 2016, doi: 10.1007/s11071-015-2481-7.

[2] L. Huang, S. Cai, X. Xiong, and M. Xiao, "On symmetric color image encryption system with permutation-diffusion simultaneous operation," Optics and Lasers in Engineering, vol. 115, pp. 7–20, 2019, doi: 10.1016/j.optlaseng.2018.11.015.

[3] Y. Luo, R. Zhou, J. Liu, Y. Cao, and X. Ding, "A parallel image encryption algorithm based on the piecewise linear chaotic map and hyper-chaotic map," Nonlinear Dynamics, vol. 93, pp. 1165–1181, 2018, doi: 10.1007/s11071-018-4251-9.

[4] Z. Hua, Y. Zhou, and H. Huang, "Cosine-transform-based chaotic system for image encryption," Information Sciences, vol. 480, pp. 403–419, 2019, doi: 10.1016/j.ins.2018.12.048.

[5] Y. Luo, S. Tang, X. Qin, L. Cao, F. Jiang, and J. Liu, "A double-image encryption scheme based on amplitude-phase encoding and discrete complex random transformation," IEEE Access, vol. 6, pp. 77740–77753, 2018, doi: 10.1109/ACCESS.2018.2884013.

[6] C. Zhu and K. Sun, "Cryptanalyzing and Improving a Novel Color Image Encryption Algorithm Using RT-Enhanced Chaotic Tent Maps," IEEE Access, vol. 6, pp. 18759–18770, 2018, doi: 10.1109/ACCESS.2018.2817600.

[7] J. Wu, X. Liao, and B. Yang, "Color image encryption based on chaotic systems and elliptic curve ElGamal scheme," Signal Processing, vol. 141, pp. 109–124, 2017, doi: 10.1016/j.sigpro.2017.04.006.

[8] U. Hayat and N. A. Azam, "A novel image encryption scheme based on an elliptic curve," Signal Processing, vol. 155, pp. 391–402, 2019, doi: 10.1016/j.sigpro.2018.10.011.

[9] Y. Luo, X. Ouyang, J. Liu, and L. Cao, "An Image Encryption Method Based on Elliptic Curve Elgamal Encryption and Chaotic Systems," IEEE Access, vol. 7, pp. 38507–38522, 2019, doi: 10.1109/ACCESS.2019.2906052.

[10] A. Banik, Z. Shamsi, and D. S. Laiphrakpam, "An encryption scheme for securing multiple medical images," Journal of Information Security and Applications, vol. 49, p. 102398, 2019, doi: 10.1016/j.jisa.2019.102398.

[11] H. I. Hussein and W. M. Abduallah, "An efficient ElGamal cryptosystem scheme," International Journal of Computers and Applications, vol. 43, no. 10, pp. 1088–1094, 2019, doi: 10.1080/1206212X.2019.1678799.

[12] A. J. Ordonez, B. D. Gerardo, and R. P. Medina, "Digital signature with multiple signatories based on modified ElGamal Cryptosystem," Proceedings of 2018 5th International Conference on Business and Industrial Research: Smart Technology for Next Generation of Information, Engineering, Business and Social Science, ICBIR 2018, pp. 89–94, 2018, doi: 10.1109/ICBIR.2018.8391172.

[13] M. Thangavel and P. Varalakshmi, "Enhanced DNA and ElGamal cryptosystem for secure data storage and retrieval in cloud," Cluster Computing, vol. 21, pp. 1411–1437, 2018, doi: 10.1007/s10586-017-1368-4.

[14] R. I. Abdelfatah, "Secure Image Transmission Using Chaotic-Enhanced Elliptic Curve Cryptography," IEEE Access, vol. 8, pp. 3875–3890, 2020, doi: 10.1109/ACCESS.2019.2958336.

[15] P.-Y. Wan, T.-L. Liao, J.-J. Yan, and H.-H. Tsai, "Discrete sliding mode control for chaos synchronization and its application to an improved El-Gamal cryptosystem," Symmetry, vol. 11, no. 7, pp. 1–13, 2019, doi: 10.3390/sym11070843.

Authorized licensed use limited to: Kent State University Libraries. Downloaded on April 26,2022 at 10:37:40 UTC from IEEE Xplore. Restrictions apply.
View publication stats