cryptography

MDPI

Article

A New Chaotic-Based RGB Image Encryption Technique Using a Nonlinear Rotational 16 × 16 DNA Playfair Matrix

Dina Ibrahim [1], Kareem Ahmed [2,*], Mohamed Abdallah [3] and AbdElmgeid A. Ali [3]

[1] Department of Computer Science, NUB University, Beni Suef 2717208, Egypt; dina.ibrahim@nub.edu.eg
[2] Department of Computer Science, Beni-Suef University, Beni Suef 2722165, Egypt
[3] Department of Computer Science, Minia University, Minya 2431436, Egypt; mas_m4@mu.edu.eg (M.A.); abdelmgeid@yahoo.com (A.A.A.)
* Correspondence: kareem_ahmed@hotmail.co.uk or kareem_ahmed@eng.bsu.edu.eg

Abstract: Due to great interest in the secure storage and transmission of color images, the necessity for an efficient and robust RGB image encryption technique has grown. RGB image encryption ensures the confidentiality of color images during storage and transmission. In the literature, a large number of chaotic-based image encryption techniques have been proposed, but there is still a need for a robust, efficient and secure technique against different kinds of attacks. In this paper, a novel RGB image encryption technique is proposed for encrypting individual pixels of RGB images using chaotic systems and 16 rounds of DNA encoding, transpositions and substitutions. First, round keys are generated randomly using a logistic chaotic function. Then, these keys are used across different rounds to alter individual pixels using a nonlinear randomly generated 16 × 16 DNA Playfair matrix. Experimental results show the robustness of the proposed technique against most attacks while reducing the consumed time for encryption and decryption. The quantitative metrics show the ability of the proposed technique to maintain reference evaluation values while resisting statistical and differential attacks. The obtained horizontal, vertical and diagonal correlation is less than 0.01, and the NPCR and UACI are larger than 0.99 and 0.33, respectively. Finally, NIST analysis is presented to evaluate the randomness of the proposed technique.

Keywords: DNA cryptography; image encryption; chaotic; Playfair; NIST analysis

Citation: Ibrahim, D.; Ahmed, K.; Abdallah, M.; Ali, A.A. A New Chaotic-Based RGB Image Encryption Technique Using a Nonlinear Rotational 16 × 16 DNA Playfair Matrix. *Cryptography* **2022**, *6*, 28. https://doi.org/10.3390/cryptography6020028

Academic Editor: Josef Pieprzyk

Received: 8 May 2022
Accepted: 5 June 2022
Published: 8 June 2022

1. Introduction

Information security is one of the most important fields that has been gaining more attention for data protection, privacy preservation and data leakage prevention. The process of securing information systems includes monitoring and analyzing all data flows and behaviors of different components that comprise the system, such as people that interact with the system; hardware, such as network infrastructure, routers and domain name servers; and software configurations and keys. It is a necessity to ensure that the system is fully functioning while protecting its components from known and unknown threats. The basic requirement for any security system is to achieve the following requirements: integrity, confidentiality and availability. Other requirements also include accountability and authentication. Encryption, also known as cryptography, is the most widely used approach to achieve confidentiality and verify integrity. It is also the main way to secure governments, social data and personal transactions, since all of these entities require a great amount of protection against known/unknown attacks and threats. In this sense, cryptography is the most fundamental building block in the design of secure information systems. As a result, it has attracted the main interest of many researchers, allowing thousands of competing research articles to offer new ways for fast and robust encryption.

Cryptography is divided into two broad categories: symmetric cryptography and asymmetric cryptography. Symmetric cryptography uses a single key for both encryption and decryption, also known as single-key cryptography. Some examples are the

Advanced Encryption Standard (AES), Data Encryption Standard (DES), 3DES and Rivest Cipher 4 (RC4). Asymmetric cryptography, also known as public-key cryptography, uses one key for encryption and another key for decryption. Some examples of public key cryptography are the Rivest–Shamir–Adleman (RSA) algorithm, Elliptic Curve Cryptography (ECC) and the Diffie–Hellman key agreement protocol.

Because of the wide spread of wireless networks, LANs, Bluetooth, and other forms of communication media, digital images should be protected to maintain privacy and copyrights. Images may be vulnerable to different types of attacks during transmission or exchange between authenticated users. Some common attacks related to networks are network denial, masquerading and eavesdropping. For any communication medium to be trusted between parties, it should satisfy reliability, anonymity, confidentiality, integrity and availability. When exchanging digital images, common ways for securing these images are cryptography and steganography. Cryptography not only guarantees confidentiality, but it can also address other issues such as data integrity, authentication and non-repudiation. With cryptography, information can be sent securely, and only the authenticated receiver can retrieve this information, thus avoiding the compromising or leakage of private data.

The RGB model is a widely used model for representing and storing digital color images. In this model, the image is represented by three independent channels. Each channel is a 2D matrix. The first matrix represents the red color, the second matrix represents the green color and the third matrix represents the blue color. The three matrices are of the same width and height. Each pixel in the color image is represented by three values that are the three corresponding values on the three matrices. Each value can range from 0 to 255. In this way, the RGB model can represent more than 16 million distinct colors. Many recent scientific papers have shown that combining DNA cryptography and chaotic sequences leads to very efficient and promising results with respect to nonlinearity, randomness and resistance to common attacks. Arthi et al. [1] presented an encryption technique based on 4D Lorenz hyper-chaos and DNA encoding. First, an integer wavelet transform IWT is applied to produce approximation and detail bands of the input image. Then, the LL band is permuted using chaotic sequences, and then they are encoded using DNA. Finally, an operation called DNA-XOR is performed to produce the encrypted image. Paul [2] presented a color image encryption technique that uses a 2D hyperchaotic map to scramble the initial image and then uses a logistic-tent map to produce a cover image. Then, SHA-2 is applied to this cover image to produce a mask image, and the two images are encoded using DNA, followed by many diffusions to produce cipher image.

In this paper, a novel RGB image encryption technique is proposed, based on a logistic chaotic function and several rounds of DNA encoding. First, the logistic function is used to generate 16 randomly generated round keys. The chaotic sequences are then employed in a Feistel structure over different rounds to change individual pixels by performing key-based random nonlinear DNA encoding, transpositions, diffusions and substitutions. An initial 16×16 DNA matrix is built up, where the entries of this matrix are four sequential DNA symbols. The matrix is first permutated according to the round key, and then Playfair is used to perform the substitution of each four DNA symbols from the bit representation of the image with the corresponding patterns, according to the common rules of Playfair. The nonlinearity of the generated matrix ensures the robustness of the technique against differential attacks. The basic Playfair substitution is preceded and followed by several transpositions and substitutions to increase the randomness of the intermediate representations. To decrypt, the encryption steps are repeated in reverse.

The rest of this paper is organized as follows. Section 2 presents an overview of the most influential techniques in the literature of DNA encryption; Section 3 explains related works with DNA cryptography and logistic chaotic functions; Section 4 explains the proposed chaotic-based RGB image encryption technique using nonlinear rotational S-Boxes of DNA codons; Section 5 presents experimental results and analytical comparisons with those that are state-of-the-art; and finally, a conclusion is presented in Section 6.

2. Literature Review

Incorporating DNA encoding into cryptography has encouraged many researchers to determine different scientific techniques for DNA cryptography. There are a huge number of techniques that use DNA to achieve security requirements while maintaining encryption and decryption speed. In this section, several recent papers related to DNA cryptography and chaotic systems are explored. Xue et al. [3] summarized DNA cryptographic techniques into five broad categories: DNA fixed coding, DNA dynamic coding, different types of base complement operations, different DNA sequence algebraic operations and combinations of multiple DNA operations. Then, they proposed a new encryption technique by combining the optimal coding mechanism with the optimal DNA coding operation. Zhang et al. [4] proposed a new image encryption technique based on DNA sequence addition operations and chaos. Kumar et al. [5] presented a DNA cryptography technique for RGB images based on the work of Lorenz and Rossler, involving a chaotic system and a 2D logistic map. Zhentao et al. [6] presented an RGB image encryption technique based on dynamic DNA and 4D memristive hyper-chaos. After generating chaotic matrices, RGB channels are encoded using dynamic DNA, and then dynamic confusion and diffusion are applied. Chen [7] proposed a new technique for DNA-based molecular cryptography. Gehani et al. [8] presented DNA cryptography using one-time pads. It works in two ways: the first is a substitution method using a dictionary of distinct randomly generated pads for a pair-wise mapping, and the second is an XOR scheme using molecular computation and a randomly generated key. These techniques are used to encode natural DNA as well as to manually encode DNA from binary data. Guesmi et al. [9] suggested a new medical image encryption technique depending on a hybrid model of DNA masking, SHA-2 and a new hybrid chaotic map. Wu et al. [10] proposed a new image encryption technique based on a two-dimensional Hénon–Sine map (2D-HSM) and the DNA approach. They found that applying DNA cryptography can greatly improve the efficiency of image permutation and diffusion. Wang et al. [11] presented an image encryption technique based on a four-wing chaotic system combined with compressed sensing (CS) and DNA coding. They used CS to reduce the image, and then they combined the Kronecker product (KP) and the chaotic system to produce a high dimensional seed matrix, which is used to control the DNA coding and is used in the XOR operation. Xiuli et al. [12] proposed a color image encryption technique based on an improved genetic algorithm and a matrix semi-tensor product (STP). The encryption is accomplished in five stages: pre-processing, DNA encoding, crossover, mutation and DNA decoding. Lu et al. [13] presented an image encryption technique based on a compound chaotic map and single S-Box. They proposed a new discrete compound chaotic system called the Logistic–Sine system (LSS), which is used to construct an S-Box. Then, they used that S-box to perform a round of permutations and two rounds of substitution. Zhang and Hu [14] proposed a multiple-image encryption algorithm based on a 3D scrambling model and dynamic DNA coding, in which they combined a generalized version of Zigzag transformation and the two-dimensional Henon–Sine map (2D-HSM). The method is able to encrypt multiple images at once. Liu and Zhang [15] proposed an image encryption technique based on multidimensional chaotic and DNA coding. Cui et al. [16] proposed a new image encryption technique based on DNA dynamic encoding and a hyper-chaotic system, in which they used SHA-3 and other processing to generate an S-box for use in encryption. Wang et al. [17] proposed a color image encryption technique based on Fisher–Yates scrambling and four DNA operations, called the DNA subsequence elongation operation, DNA subsequence truncation operation, DNA subsequence deletion operation and DNA subsequence insertion arithmetic. They used a Chen system to generate chaotic sequences, and then they used the Fisher–Yates scrambling method to encrypt red, green and blue channels. Huang and Zhou [18] proposed an image encryption technique that combines a 6D non-degenerate discrete hyperchaotic system and a 2D discrete cosine Stockwell transform with DNA-level modulus diffusion.

3. Related Work

3.1. DNA Cryptography

DNA cryptography is considered one of the most appealing directions in cryptography. Huge amounts of parallelism and exceptional data thickness make the attributes of DNA suitable for being used in cryptography applications, such as encryption, confirmation and digital signatures [19]. DNA cryptography is a broad category in cryptography, which involves improved encoding and a great amount of randomness. DNA cryptography uses some encoding techniques to convert binary streams into DNA sequences, and then it uses different concepts related to DNA, such as nucleotides, proteins, amino acids and complementary rules to perform encryption, decryption and key generation. DNA steganography is another interesting field, in which secret data are embedded into the biological structure of natural DNA. First, the secret data are transformed into a DNA sequence. Then, this sequence is merged with a real DNA pattern in such a way; therefore, the output of this operation is a natural DNA sequence. DNA is available in the NCBI database (National Center for Biotechnology Information). DNA cryptography is always used along with DNA steganography to further protect sensitive data. In this case, data is first encrypted and then hidden inside a DNA sequence. DNA cryptography depends on DNA calculations, as the motive for its use is to simulate natural tasks occurring in DNA sequences.

Deoxyribonucleic acid (DNA) (https://www.msdmanuals.com/home last accessed on 7 June 2022) is a complex chemical compound found in all organisms and is of great importance because it contains all necessary information to generate all kinds of proteins in a cell. DNA consists of two linked strands that wrap around each other and is called a double helix. DNA is composed up of four base units called nucleotides: adenine (A), thymine (T), guanine (G) and cytosine (C). The nucleotides in the first strand are connected with the nucleotides in the second strand to form chemical bonds, called base pairs. Adenine (A) is connected only to thymine, and vice versa. Guanine (G) is connected only to cytosine (C), and vice versa. A gene is a short segment of DNA, and it contains specific genetic information because it contains the code for specific protein that performs some function in one or more types of cells in the human body. There is another important concept called a chromosome, which is a structure existing inside a cell's nucleus, and chromosomes contain genes. A chromosome may contain hundreds or thousands of genes.

As mentioned before, genes contain DNA. DNA is a blueprint (code) which is responsible for synthesizing a protein. Proteins are the basic building blocks for skin, muscles and other parts of the human body, but the importance of proteins is that they are responsible for making enzymes, which are responsible for all chemical processes and reactions inside the human body. The body is controlled by thousands of enzymes. Therefore, the amount and type of proteins control the functions and structures of the whole body. In this sense, the synthesis of proteins is the most important operation that affects all aspects regarding the structures and functions of the human body.

Proteins are long chains of amino acids connected to each other. The instructions for synthesizing proteins are stored in DNA, where nucleotides inside DNA are arranged in groups. Each group contains three consecutive nucleotides. Each group is also called a codon, and a codon is a group of three adjacent nucleotides. Each codon is considered an instruction to add a specific amino acid to the protein chain. For example, the GCT codon is an instruction to add the amino acid alanine. Therefore, the sequence of amino acids inside a protein chain is specified by the order of the codons in the gene for that protein. The process of turning DNA into a protein is caried out by two subprocesses called transcription and translation. In transcription, the instructions in DNA are copied (transcribed) to ribonucleic acid (RNA). RNA is similar to a strand of DNA, but thymine (T) is replaced by uracil (U). RNA typically contains codons of three bases, similar to DNA. Table 1 shows the binary representations of mRNA codons. In translation, the mRNA that is obtained from DNA is translated into an amino acid [20]. The main differences between DNA and RNA are that the sugar in DNA is deoxyribose, whereas the sugar in RNA is

ribose. Moreover, RNA uses uracil (U) instead of thymine (T). RNA codons are codons that are used during a process called transcription, but each mRNA molecule acquires its sequence of nucleotides by transcription from the corresponding gene of DNA.

Table 1. Binary representations of mRNA Codons.

	A	**U**	**C**	**G**	
A [00]	AAA [000000]	AUA [000100]	ACA [001000]	AGA [001100]	A
	AAU [000001]	AUU [000101]	ACU [001001]	AGU [001101]	U
	AAC [000010]	AUC [000110]	ACC [001010]	AGC [001110]	C
	AAG [000011]	AUG [000100]	ACG [001011]	AGG [001111]	G
U [01]	UAA [010000]	UUA [010100]	UCA [011000]	UGA [011100]	A
	UAU [010001]	UUU [010101]	UCU [011001]	UGU [011101]	U
	UAC [010010]	UUC [010110]	UCC [011010]	UGC [011110]	C
	UAG [010011]	UUG [010111]	UCG [011011]	UGG [011111]	G
C [10]	CAA [100000]	CUA [100100]	CCA [101000]	CGA [101100]	A
	CAU [100001]	CUU [100101]	CCU [101001]	CGU [101101]	U
	CAC [100010]	CUC [100110]	CCC [101010]	CGC [101110]	C
	CAG [100011]	CUG [100111]	CCG [101011]	CGG [101111]	G
G [11]	GAA [110000]	GUA [110100]	GCA [111000]	GGA [111100]	A
	GAU [110001]	GUU [110101]	GCU [111001]	GGU [111101]	U
	GAC [110010]	GUC [110110]	GCC [111010]	GGC [111110]	C
	GAG [110011]	GUG [110111]	GCG [111011]	GGG [111111]	G

This process occurs in the nucleus. In this process, information in DNA is copied into the mRNA. This process is conducted in three stages. (1) Initiation: This phase occurs when the enzyme RNA polymer is associated with DNA in a place called a promoter. Consequently, the enzyme is able to read DNA in one part and form the RNA strand. (2) Elongation: At this stage, the RNA strand elongates by adding nucleotides. The RNA polymer reads a DNA part and forms mRNA using a complementary base, and mRNA uses uracil (U) instead of thymine (T). (3) Termination: This is the last stage of transcription and occurs when the mRNA is completed and separated from the DNA.

A genetic code can be defined as the relationship between the nucleotide sequences of genes and the amino acid sequences of proteins, which is determined by the rules of translation. It consists of three-letter 'words' called codons, formed from a sequence of three nucleotides (e.g., UCA or GAC). Each codon specifies one amino acid in a protein. It can be expressed as either a DNA codon or an RNA codon. The process of protein production or translation occurs after the transcription process. Table 2 shows all possible codons and their corresponding amino acids. There are 20 different amino acids that can be used in protein synthesis. The protein-building process is stopped when the process arrives at the three "stop" codons (UAG, UAA, UGA) [7].

The rest of this section presents one of the most used rules in DNA cryptography, which is the DNA complementary base pairing rule. As mentioned before, DNA consists of two strands of nucleotides called a double helix. The nucleotides are cytosine (C), guanine (G), adenine (A) or thymine (T). They are bound by hydrogen bonds. For example, A with T includes two hydrogen bonds, and C with G includes three hydrogen bonds. This chemical structure of the bases is called the Watson–Crick complementary base pairing principle. Arranging these rules helps to calculate and store information. The encoding rule is the following: (A-00), (C-01), (G-10) and (T-11), and the complementary numbers of them are (00-11), (01-10), (10-01) and (11-00). There are eight cases of combinations that satisfy complementary base pairing, as shown in Table 3 [21].

Table 2. Sixty-four different codons and their corresponding amino acid [20].

		U		**C**		**A**		**G**	
U	U	UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys
	C	UUC		UCC		UAC		UGC	
	A	UUA	Leu	UCA		UAA	Stop	UGA	Stop
	G	UUG		UCG		UAG		UGG	Trp
C	U	CUU	Leu	CCU	Pro	CAU	His	CGU	Arg
	C	CUC		CCC		CAC		CGC	
	A	CUA		CCA		CAA	Gln	CGA	
	G	CUG		CCG		CAG		CGG	
A	U	AUU	Ile	ACU	Thr	AAU	Asn	AGU	Ser
	C	AUC		ACC		AAC		AGC	
	A	AUA		ACA		AAA	Lys	AGA	Arg
	G	AUG	Met	ACG		AAG		AGG	
G	U	GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly
	C	GUC		GCC		GAC		GGC	
	A	GUA		GCA		GAA	Glu	GGA	
	G	GUG		GCG		GAG		GGG	

Table 3. DNA encoding rules.

Rule	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
00	T	T	G	C	G	C	A	A
01	G	C	T	T	A	A	G	C
10	C	G	A	A	T	T	C	G
11	A	A	C	G	C	G	T	T

3.2. Chaotic System

A chaotic system is a system that performs different actions in an unpredictable manner. It seems to perform random behavior, but it actually is a deterministic behavior predetermined by mathematical equations. The system is controlled by nonlinear dynamic systems with certain conditions. These dynamic systems are very sensitive to the initial conditions provided by chaotic maps. There are many chaotic maps that can generate chaotic motion of a single dimension. Sridevi et al. [22] presented a color image encryption technique based on chaos, which uses confusion and diffusion based on chaotic maps. They used a logistic map, the Lorenz Attractor, a Tent map and the Lu attractor with different initial conditions and different initial values. Liu et al. [23] proposed a color image encryption technique based on a diffusion matrix generated from the Hopfield chaotic neural network (HCNN). Wu et al. [10] proposed a two-dimensional Hénon–Sine map (2D-HSM), which has better ergodicity and pseudo randomness. Joshi et al. [24] proposed a new image encryption technique using the 2D multiple parameter fractional discrete Fourier transform, a baker chaotic map and the 3D Arnold transform.

In this subsection, we present the most usable chaotic function, which is widely used to achieve complete randomness within chaotic sequences [25]. A logistic map is a chaos map that creates a chaotic signal of random sequences and is extremely sensitive to the previous sequence and the initial parameter λ. A logistic one-dimensional map with an unexpected degree of complexity is defined in Equation (1).

$$x_{n+1} = \lambda x_n(1 - x_n) \tag{1}$$

where λ controls the degree of randomness by folding or stretching the interval of generated sequences and $0 < \lambda \leq 4$. All sequences are generated with a great amount of randomness. The value of λ is always set to a value greater than 3.57 to show great chaotic behavior. All generated sequences are in the interval $0 < x_i < 1$. It is impossible to detect the convergence of iterative values after any number of repetitions [26]. A graph of a logistic

map shows folding and stretching of values between 0 and 1, and for λ, it shows values between 3.47 and 4. In this sense, randomly generated sequences are suitable for many operations, such as key generation.

4. General Framework of the Proposed Strategy

After presenting the fundamentals of DNA cryptography, complementary rules and the logistic chaotic map, we introduce our algorithm based on DNA cryptography. Before encryption, the communication parties should provide a secure channel for exchanging the secret key between the sender and the receiver. Therefore, if an attacker is listening to the communication, he is still unable to determine the key [20]. They can use any key exchange algorithm, such as Diffie–Hellman to exchange the secret key. Image encryption is considered one of the most important techniques used for protecting images from unauthorized view or modification. Recently, deoxyribonucleic acid (DNA) technology has had an influence on several sectors, including the medical system, cryptography, computing and other fields in information technology. DNA is a grouping of four nucleic bases: A (adenine), C (cytosine), G (guanine) and T (thymine). The image is divided into three color channels: red (R), green (G) and blue (B). The data from these three channels are encoded in binary matrices. These matrices are then encoded using DNA encoding principles. DNA operations are used on the encoded matrices to remove similarities between pixel values. After that, decoding methods are used to transform them back into binary matrices. Finally, to create a cipher-colored image, three color channels are converted to decimal and are joined again. Chaotic maps are widely used in secure communication and are seen to be a good compromise between security and computing efficiency. A chaos-based encryption method has various characteristics, including sensitivity to initial values and parameters selected. True randomness of the generated patterns and accurate values are regenerated if the same parameters and initial values are provided [27].

In this section, we go step by step to illustrate the proposed RGB image encryption technique. An RGB image is shown by a linear set of three unique channels, which are R (Red), G (Green) and B (Blue), as shown in Figure 1. In our algorithm, we go through 16 rounds to produce the encrypted image; therefore, it becomes practically impossible for an attacker to figure it out by brute force or other known attacks. It is important for any conventional encryption technique to be built based on the Feistel structure. Feistel proposed utilizing transpositions and substitutions interchangeably across several rounds. The encryption calculation should be reversible, which implies that it can scramble the plain content and unscramble the obtained cipher. The proposed algorithm consists of 16 rounds, and each round consists of a set of random substitutions and transpositions defined in the following subsections. The last key obtained after each round is used as the starting key for the next round. The pseudo code for the proposed algorithm is as follows:

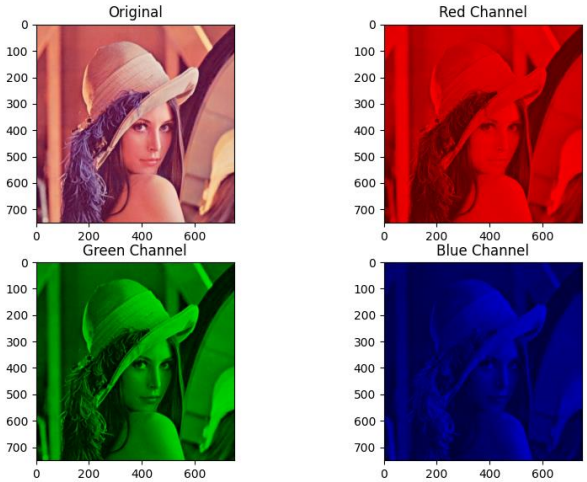

Figure 1. Original Lena image, Red Green and Blue Channels of Lena image.

Input:

1. Original RGB image I of size $M \times N$
2. 16-character (128 bit) key K

Output: Encrypted Image I'

Encryption Algorithm:

1. Use a logistic map to generate 15 round keys as described in Section 4.1.
2. Repeat steps 3–15 sixteen times, using one key from the 16 keys (Initial key + 15 generated keys) in each round. A round includes the execution of all steps from 3 to 15.
3. Convert red, green and blue channels into three vectors, *vec_R*, *vec_G* and *vec_B*, of size $(1, M \times N)$.
4. Circular left shift *vec_R* by the decimal value of $K(0:15)$.
5. Circular left shift *vec_G* by the decimal value of $K(16:31)$.
6. Circular left shift *vec_B* by the decimal value of $K(32:47)$.
7. Prepare the 6×4 final map as described in Section 4.2 using $K(0:23)$.
8. For each pixel $P = (P_r, P_g, P_b)$ in the obtained image after circular shifts of individual channel vectors:
   - 8.1. Convert P_r, P_g, P_b into their 8-bit binary representations $Bits(P_r)$, $Bits(P_g)$, $Bits(P_b)$.
   - 8.2. Construct a 24-bit binary string S by concatenating $Bits(P_r)$, $Bits(P_g)$, $Bits(P_b)$.
   - 8.3. Permutate the string S according to the 6×4 final map obtained from step 7.
   - 8.4. Update the pixel values in *vec_R*, *vec_G* and *vec_B*.
9. Construct an initial 16×16 DNA Playfair matrix of all possible combinations of 4-DNA nucleotides, starting from AAAA to TTTT as described in Section 4.3.
10. Divide the 128-bit round key into two halves, each of which is 64 bits. Use the left half to permutate the rows of the 16×16 DNA Playfair matrix, and use the right half to permutate the columns of the 16×16 DNA Playfair matrix as described in Section 4.3.
11. If the round index is odd, then pick 1 pixel from the red vector and 1 pixel from the green vector in the forward direction; otherwise, pick 2 pixels from the red and green vectors in the backwards direction.
12. Apply the modified Playfair procedure as described in Section 4.3 to substitute the 2 pixels' values with their corresponding values.
13. After encrypting 8 pixels in the red vector and 8 pixels in the green vector, combine the encrypted pixels in 1 vector, the 8 encrypted pixels in the red vector, followed by the 8 encrypted pixels in the green vector. Then, perform the XOR operation between the current key and the 128 bits of the 16 pixels, as described in Section 4.5.
14. Repeat steps 10–13 until all pixels in the red vector and green vectors are substituted; consider the direction based on the round index.
15. Encrypt the blue channel using the complementary rules in Table 3 as described in Section 4.4.

For decryption, the above stages are performed in reverse after generating the same 15 keys plus the initial key, and the last key $K15$ is used first and applied to the pixels in the backwards direction. The XOR operation with the current key is performed directly with the encrypted pixels in the red and green vectors before performing decoding using the 16×16 DNA matrix. Finally, the 6×4 matrix is used for rearrangement.

4.1. Round Key Generation

In our algorithm, an initial key of 128 bits is entered. This key is used to generate 15 round keys, each of which is 128 bits. Sixteen chaotic logistic functions are used to generate each key, as described in Table 4. The input key is used for the first round and is used to generate another key which is used for the second round. Then, each round key is used to scramble the temporary image at this round and is also used to generate the next key. The procedure works as follows:

Table 4. Using a logistic chaotic map to calculate the second round key.

$K[i-j]$	8 Bits	Decimal	Decimal/256	Initial Value	Logistic Function	Obtained Value	Decimal	Mod 256	Bin
$K[0-7]$	01000100	68	0.265625	0.265	0.77715225	0.777	777	9	00001001
$K[8-15]$	01101001	105	0.41015625	0.41	0.965181	0.965	965	197	11000101
$K[16-8]$	01101110	110	0.4296875	0.429	0.97738641	0.977	977	209	11010001
$K[24-31]$	01100001	97	0.37890625	0.378	0.93811284	0.938	938	170	10101010
$K[32-9]$	01000101	69	0.26953125	0.269	0.78458961	0.784	784	16	00010000
$K[40-47]$	01101100	108	0.421875	0.421	0.97259841	0.972	972	204	11001100
$K[48-10]$	01100010	98	0.3828125	0.382	0.94194324	0.941	941	173	10101101
$K[56-63]$	01100001	97	0.37890625	0.378	0.93811284	0.938	938	170	10101010
$K[64-11]$	01101110	110	0.4296875	0.429	0.97738641	0.977	977	209	11010001
$K[72-79]$	01100001	97	0.37890625	0.378	0.93811284	0.938	938	170	10101010
$K[80-12]$	00110001	49	0.19140625	0.191	0.61653081	0.616	616	104	01101000
$K[88-95]$	00110010	50	0.1953125	0.195	0.62633025	0.626	626	114	01110010
$K[96-13]$	00110011	51	0.19921875	0.199	0.63600201	0.636	636	124	01111100
$K[104-111]$	00110100	52	0.203125	0.203	0.64554609	0.645	645	133	10000101
$K[112-14]$	00110101	53	0.20703125	0.207	0.65496249	0.654	654	142	10001110
$K[120-127]$	00110110	54	0.2109375	0.21	0.661941	0.661	661	149	10010101

i. The current key is divided into 16 groups, each of which is 8 bits.
ii. Each group is converted to a decimal value.
iii. Each decimal value is divided by 256.
iv. For each decimal value obtained in the previous step, the first three decimal places are kept, and the remaining digits are simply chopped off. The result is used as the initial value for the logistic function.
v. The logistic function defined by Equation (1) is applied to each three-digit decimal value, with $\lambda = 3.99$ to increase the randomness of the chaotic sequence.
vi. For each decimal value obtained in the previous step, the first three decimal places are kept, and the remaining digits are simply chopped off.
vii. The result of each group is multiplied by 1000 to remove all decimal digits and to obtain an integer value.
viii. The remainder of dividing the obtained integer value by 256 is calculated for each group.
ix. The obtained remainder is converted back to an 8-bit binary value.
x. The first 8 binary groups are merged together to form a 64-bit left key, and the second 8 binary groups are merged together to form a 64-bit right key. Combining the groups in Table 4 produces the following: left = 0000100111000101110100011010101000010000110011001010110110101010 right = 1101000110101010011010000111001001111100100001011000111010010101
xi. Perform one left shift on each part, then combine the result into a single key. In the previous example, key = 0001001110001011101000110101010000100001100110010101101101010100101000110101010011010000111001001111100100001011000111010010101011
xii. Repeat steps *i-x* on the obtained key, and combine all binary groups together to produce the final 128-bit round key.

4.2. Initial Permutation of Merged Bit String

In this section, we explain the basic steps for performing the initial permutation of the input RGB image using the key. To make things simple and clear, we go step by step in the first round, where the proposed approach accepts an RGB image and a 128-bit secret

key. Let the secret key = "*DinaElbana*123456". The first eight steps of our algorithm are related to random permutations of the pixels' bits. These steps are repeated 16 times, 1 at the beginning of each round. At each round, the round key is used, which is generated according to the logistic function defined in Equation (1) as described in Section 4.1. The first step is to extract the three color channels from the RGB image, which are the red channel R, the green channel G and the blue channel B, as shown in Figure 1. These channels are considered three matrices, and each one has the same number of rows and columns. Each matrix of these matrices is converted into its corresponding vector.

Let the three channels be defined as follows:

$$RedChannelMatrix \begin{bmatrix} 98 & 135 & 180 & 215 \\ 95 & 209 & 254 & 19 \\ 160 & 201 & 56 & 127 \\ 83 & 145 & 227 & 28 \end{bmatrix}$$

$$GreenChannelMatri \begin{bmatrix} 180 & 168 & 254 & 211 \\ 207 & 51 & 156 & 15 \\ 129 & 101 & 209 & 98 \\ 165 & 75 & 6 & 245 \end{bmatrix}$$

$$BlueChannelMatri \begin{bmatrix} 221 & 159 & 118 & 222 \\ 101 & 185 & 19 & 250 \\ 57 & 25 & 84 & 205 \\ 76 & 65 & 44 & 157 \end{bmatrix}$$

In this sense, the red channel vector vec_R equals [98, 135, 180, 215, 95, 209, 254, 19, 160, 201, 56, 127, 83, 145, 227, 28].

The green channel vector vec_G equals [6, 15, 51, 75, 98, 101, 129, 156, 165, 168, 180, 207, 209, 211, 245, 254].

The blue channel vector vec_B equals [221, 159, 118, 222, 101, 185, 19, 250, 57, 25, 84, 205, 76, 65, 44, 157].

The second step is to convert the 16-character key into its binary 128-bit equivalent. In our case, we choose the key "*DinaElbana*123456", so the bit representation is as follows:

$Key =$

01000100011010010110111001100001
01000101011011000110001001100001
01101110011000010011000100110010
00110011001101000011010100110110

Next, the first 16 bits are extracted from the key and assigned to Red_shifts. The second 16 bits are extracted and assigned to $Green\_shifts$, and the third 16 bits are extracted and assigned to $Blue\_shifts$. Then, all these bits are converted into their decimal values. In our case:

$$Red\_shifts = 0100010001101001 = (17513)$$

$$Green\_shifts = 0110111001100001 = (28257)$$

$$Blue\_shifts = 0100010101101100 = (17772)$$

Next, we use these values to determine the amount of circular left shifts that we should perform on the red vector, green vector and blue vector. Different kinds of shifts (right or left) may be performed to increase the amount of randomness. However, you should remember the order in which different kinds of shifts have been made to be able to decrypt the cipher image. The rotation is performed by the obtained decimal values. When applying to previous vectors corresponding to 4×4 matrices, it seems as though the shifts are being repeated too much, but regarding an actual image of size 512×512, for example, the vector is $262,144$, which is larger than the maximum possible number of shifts, which is $65,536$. When applying the previous step to the three vectors, the red vector vec_R is shifted left by $17,513$, the green vector vec_G is shifted left by $28,257$ and the blue vector

vec_B is shifted left by 17,772. The results are a random arrangement of pixels on the three independent vectors.

The red channel vector *vec_R* is $[19, 28, 56, 83, 95, 98, 127, 135, 145, 160, 180, 201, 209, 215, 227, 254]$.

The green channel vector *vec_G* is $[6, 15, 51, 75, 98, 101, 129, 156, 165, 168, 180, 207, 209, 211, 245, 254]$.

The blue channel vector *vec_B* is $[19, 25, 44, 57, 65, 76, 84, 101, 118, 157, 159, 185, 205, 221, 222, 250]$.

The last step is to construct a random matrix of 6×4, which contains numbers from 0 to 23 as follows:

$$Map = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \end{bmatrix}$$

Taking the first 24 bits from the key, bits from 0 to 11 determine the number of left shifts to the rows, and bits from 12 to 23 determine the number of up shifts to the columns. The operation is conducted as follows:

i. Bits $0, 1$ determine the number of left shifts to the first row.
ii. Bits $2, 3$ determine the number of left shifts to the second row.
iii. Bits $4, 5$ determine the number of left shifts to the third row.
iv. Bits $6, 7$ determine the number of left shifts to the fourth row.
v. Bits $8, 9$ determine the number of left shifts to the fifth row.
vi. Bits $10, 11$ determines number of left shifts to the sixth row.
vii. Bits $12 - 14$ determine the number of up shifts to the first column.
viii. Bits $15 - 17$ determine the number of up shifts to the second column.
ix. Bits $18 - 20$ determine the number of up shifts to the third column.
x. Bits $21 - 23$ determine the number of up shifts to the fourth column.

After determining the number of rolls at each row and column, the rolling is performed according to the following procedure:

i. The first row is rolled left by $K[0:1]$, and then the first column is rolled up by $K[12:14]$.
ii. The second row is rolled left by $K[2:3]$, and then the second column is rolled up by $K[15:17]$.
iii. The third row is rolled left by $K[4:5]$.
iv. The fourth row is rolled left by $K[6:7]$.
v. The fifth row is rolled left by $K[8:9]$, and then the third column is rolled up by $K[18:20]$.
vi. The sixth row is rolled left by $K[10:11]$, and then the fourth column is rolled up by $K[21:23]$.

In our example the first 24 bits of the key are 010001000110100101101110, so we perform the following:

i. The first row is rolled left by $(01)_b = 1$, and then the first column is rolled up by $(100)_b = 4$.
ii. The second row is rolled left by $(00)_b = 0$, and then the second column is rolled up by $(101)_b = 5$.
iii. The third row is rolled left by $(01)_b = 1$.
iv. The fourth row is rolled left by $(00)_b = 0$.
v. The fifth row is rolled left by $(01)_b = 1$, and then the third column is rolled up by $(101)_b = 5$.
vi. The sixth row is rolled left by $(10)_b = 2$, and then the fourth column is rolled up by $(110)_b = 6$, which is equal to rolling up by 0.

After applying these shifts, the final map is:

$$Map = \begin{bmatrix} 16 & 21 & 22 & 0 \\ 20 & 2 & 3 & 7 \\ 5 & 10 & 6 & 1 \\ 4 & 9 & 11 & 15 \\ 13 & 18 & 14 & 8 \\ 19 & 23 & 12 & 17 \end{bmatrix}$$

The final map matrix is converted into a vector that represents the permutation of 24 bits merged from the red, green and blue channels of the current pixel. The map vector is as follows:

$$Map = [16, 21, 22, 0, 20, 2, 3, 7, 5, 10, 6, 1, 4, 9, 11, 15, 13, 18, 14, 8, 19, 23, 12, 17]$$

This means that bit number 16 is the first bit, bit number 21 is the second bit, etc. For example, the pixel at the second row and second column in the original matrices for red, green and blue is $(209, 51, 185)$. Remember that we should work on already rotated vectors, but this is only for the illustration of the permutation process. Each value is converted into binary, and the 3-bit strings are merged together as follows:

$$(209)_{10} = (11010001)_2$$
$$(51)_{10} = (00110011)_2$$
$$(185)_{10} = (10111001)_2$$

The merged bit string is

$$110100010011001110111001$$

We then permute the bits according to the Map vector, and the permuted bit string is equal to

$$100110110101001101101100$$

Then, we convert each 8 bit to its corresponding decimal value as follows:

$$(10011011)_2 = (155)_{10}$$
$$(01010011)_2 = (83)_{10}$$
$$(01101100)_2 = (108)_{10}$$

Figure 2 shows the scrambled image after the first permutation in the first round. The following are the values obtained when applying a 24-bit string permutation on the original matrices to show the amount of randomness and difference obtained.

Old Red channel

$$\begin{bmatrix} 98 & 135 & 180 & 215 \\ 95 & \mathbf{209} & 254 & 19 \\ 160 & 201 & 56 & 127 \\ 83 & 145 & 227 & 28 \end{bmatrix}$$

Transposed Red channel

$$\begin{bmatrix} 204 & 249 & 118 & 251 \\ 67 & \mathbf{155} & 54 & 171 \\ 28 & 25 & 70 & 207 \\ 75 & 19 & 93 & 202 \end{bmatrix}$$

Old Green channel

$$\begin{bmatrix} 180 & 168 & 254 & 211 \\ 207 & \mathbf{51} & 156 & 15 \\ 129 & 101 & 209 & 98 \\ 165 & 75 & 6 & 245 \end{bmatrix}$$

Transposed Green channel

$$\begin{bmatrix} 114 & 224 & 198 & 183 \\ 189 & \mathbf{83} & 186 & 33 \\ 1 & 93 & 15 & 252 \\ 113 & 5 & 48 & 207 \end{bmatrix}$$

Old Blue channel

$$\begin{bmatrix} 221 & 159 & 118 & 222 \\ 101 & \mathbf{185} & 19 & 250 \\ 57 & 25 & 84 & 205 \\ 76 & 65 & 44 & 157 \end{bmatrix}$$

Transposed Blue channel

$$\begin{bmatrix} 157 & 30 & 251 & 57 \\ 247 & \mathbf{108} & 158 & 235 \\ 92 & 140 & 25 & 37 \\ 145 & 39 & 224 & 156 \end{bmatrix}$$

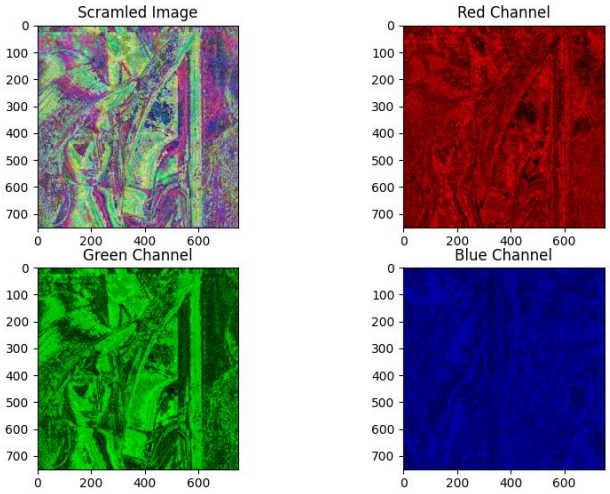

Figure 2. Scrambled image after permutation of merged bit string of RGB channels in the first round.

4.3. Nonlinear Rotational 16 × 16 *DNA S-Boxes*

The initial Playfair cipher constructs a 5 × 5 matrix which contains all alphabetic letters, with the exception of a letter $I = J$ or Q. A secret keyword is chosen, and then a 5 × 5 matrix is constructed by arranging the letters of the keyword without repetition from left to right and top to bottom, followed by the rest of letters in the alphabet [28]. The message to be encrypted is divided into diagrams or groups of two letters. When the two letters in the group are similar, another character is used as a separator and is placed between them. If the number of characters in the message is odd, a padding letter is added at the end. Substitution occurs according to the three rules listed below:

1. If the two letters are on the same row, they are replaced with the two letters to the right. When one of the letters is at the end of the row, it is replaced by the first letter in the row.
2. If the two letters are on the same column, they are replaced by the two letters below them. If one of the letters is at the end of the column, it is replaced by the first letter in the column.
3. If the two letters are not on the same row or the same column, a rectangle shape is made with letters, and the letters on the opposite corners are used to replace them.

In our proposed technique, we first construct an initial Playfair matrix of all possible combinations of four DNA nucleotides, starting from AAAA to TTTT. This constructs a 16 × 16 DNA matrix, where each entry is four DNA nucleotides, as shown in Figure 3. Rows are numbered from 0 to 15, and columns are numbered from 0 to 15.

AAAA	AAAC	AAAG	AAAT	AACA	AACC	AACG	AACT	GAAA	GAAC	GAAG	GAAT	GACA	GACC	GACG	GACT
AAGA	AAGC	AAGG	AAGT	AATA	AATC	AATG	AATT	GAGA	GAGC	GAGG	GAGT	GATA	GATC	GATG	GATT
ACAA	ACAC	ACAG	ACAT	ACCA	ACCC	ACCG	ACCT	GCAA	GCAC	GCAG	GCAT	GCCA	GCCC	GCCG	GCCT
ACGA	ACGC	ACGG	ACGT	ACTA	ACTC	ACTG	ACTT	GCGA	GCGC	GCGG	GCGT	GCTA	GCTC	GCTG	GCTT
AGAA	AGAC	AGAG	AGAT	AGCA	AGCC	AGCG	AGCT	GGAA	GGAC	GGAG	GGAT	GGCA	GGCC	GGCG	GGCT
AGGA	AGGC	AGGG	AGGT	AGTA	AGTC	AGTG	AGTT	GGGA	GGGC	GGGG	GGGT	GGTA	GGTC	GGTG	GGTT
ATAA	ATAC	ATAG	ATAT	ATCA	ATCC	ATCG	ATCT	GTAA	GTAC	GTAG	GTAT	GTCA	GTCC	GTCG	GTCT
ATGA	ATGC	ATGG	ATGT	ATTA	ATTC	ATTG	ATTT	GTGA	GTGC	GTGG	GTGT	GTTA	GTTC	GTTG	GTTT
CAAA	CAAC	CAAG	CAAT	CACA	CACC	CACG	CACT	TAAA	TAAC	TAAG	TAAT	TACA	TACC	TACG	TACT
CAGA	CAGC	CAGG	CAGT	CATA	CATC	CATG	CATT	TAGA	TAGC	TAGG	TAGT	TATA	TATC	TATG	TATT
CCAA	CCAC	CCAG	CCAT	CCCA	CCCC	CCCG	CCCT	TCAA	TCAC	TCAG	TCAT	TCCA	TCCC	TCCG	TCCT
CCGA	CCGC	CCGG	CCGT	CCTA	CCTC	CCTG	CCTT	TCGA	TCGC	TCGG	TCGT	TCTA	TCTC	TCTG	TCTT
CGAA	CGAC	CGAG	CGAT	CGCA	CGCC	CGCG	CGCT	TGAA	TGAC	TGAG	TGAT	TGCA	TGCC	TGCG	TGCT
CGGA	CGGC	CGGG	CGGT	CGTA	CGTC	CGTG	CGTT	TGGA	TGGC	TGGG	TGGT	TGTA	TGTC	TGTG	TGTT
CTAA	CTAC	CTAG	CTAT	CTCA	CTCC	CTCG	CTCT	TTAA	TTAC	TTAG	TTAT	TTCA	TTCC	TTCG	TTCT
CTGA	CTGC	CTGG	CTGT	CTTA	CTTC	CTTG	CTTT	TTGA	TTGC	TTGG	TTGT	TTTA	TTTC	TTTG	TTTT

Figure 3. Original 16 × 16 DNA Playfair matrix.

Next, we use the round key to randomly permutate the entries of the matrix defined in Figure 3. The original key consists of 128-bit, and we divide it into two parts. Each part consists of 64 bits. The first part is organized into 16 groups. Each group consists of 4-bits, and each group is a decimal value from 0 to 15. This decimal value is used to rotate one of the rows in the original Playfair matrix to the left. Similarly, the second part, which is 64-bit, is organized into 16 groups. Each group consists of 4-bits, and each group is a decimal value from 0 to 15. This decimal value is used to rotate up one of the columns in the Playfair matrix. The rows are shifted left first, then the columns are shifted up. Other variants of this proposed algorithm include shifting one row to the left followed by shifting one column up. However, remember the order on which shifts are performed to be able to restore the Playfair matrix. After applying these shifts using our initial key, the produced Playfair matrix is as is shown in Figure 4.

AACG	AACT	GAAA	GAAC	GAAG	GAAT	GACA	GACC	GTCG	GATT	AAAA	AAAC	AAAG	AAAT	AACA	AACC
GTTG	GCCT	AAGA	AAGC	AAGG	AAGT	AATA	AATC	AATG	AATT	GAGA	GAGC	GAGG	GAGT	GATA	GATC
ACCG	ACCT	GCAA	GCAC	GCAG	GCAT	GCCA	GCCC	TACG	GCTT	ACAA	ACAC	ACAG	ACAT	ACCA	ACCC
ACGC	ACGG	ACGT	ACTA	ACGA	ACTG	ACTT	GCGA	GCGC	GCGG	GCGT	GCTA	GCTC	TATG	GGCT	ACTC
AGAT	AGCA	AGCC	AGCG	AGCT	GGAA	GGAC	GGAG	GGAT	GGCA	GGCC	TCCG	GGTT	AGAA	AGAC	AGAG
AGGC	AGGG	AGGT	AGTA	AGTC	AGTG	AGTT	GGGA	GGGC	GGGG	GGGT	GGTA	GGTC	TCTG	GTCT	AGGA
ATAT	ATCA	ATCC	ATCG	ATCT	GTAA	GTAC	GTAG	GTAT	GTCA	GTCC	TGCG	GTTT	ATAA	ATAC	ATAG
ATGG	ATGT	ATTA	ATTC	ATTG	ATTT	GTGA	GTGC	GTGG	GTGT	GTTA	GTTC	TGTG	TACT	ATGA	ATGC
CAAT	TACA	CACC	CACG	CACT	TAAA	TAAC	TAAG	TAAT	CACA	TACC	TTCG	TATT	CAAA	CAAC	CAAG
CAGT	CATA	CATC	CATG	CATT	TAGA	TAGC	TAGG	TAGT	TATA	TATC	TTTG	TCCT	CAGA	CAGC	CAGG
CCAT	CCCA	CCCC	CCCG	CCCT	TCAA	TCAC	TCAG	TCAT	TCCA	TCCC	GACG	TCTT	CCAA	CCAC	CCAG
CCTA	TCTC	CCTG	CCTT	TCGA	TCGC	TCGG	TCGT	TCTA	CCTC	GATG	TGCT	CCGA	CCGC	CCGG	CCGT
CGAT	CGCA	CGCC	CGCG	CGAG	TGAA	TGAC	TGAG	TGAT	TGCA	TGCC	GCCG	TGTT	CGAA	CGAC	CGCT
CGTC	CGTG	CGTT	TGGA	TGGC	TGGG	TGGT	TGTA	TGTC	GCTG	TTCT	CGGA	CGGC	CGGG	CGGT	CGTA
CTAT	CTCA	CTCC	CTCG	CTCT	TTAA	TTAC	TTAG	TTAT	TTCA	TTCC	GGCG	TTTT	CTAA	CTAC	CTAG
CTTG	CTTT	TTGA	TTGC	TTGG	TTGT	TTTA	TTTC	GGTG	GACT	CTGA	CTGC	CTGG	CTGT	CTTA	CTTC

Figure 4. Randomly generated 16 × 16 DNA Playfair matrix.

After producing the nonlinear random 16 × 16 DNA Playfair matrix through key-based rotating operations, we use this matrix to encrypt the red and green channels. The operation is performed by taking one pixel from the red channel and the corresponding pixel in the green channel. Let the two values be 124 and 78. Each value of these pixels is eight bits, i.e., 124 corresponds to 01111100 in binary, and 78 corresponds to 01001110 in binary. Now, we make the substitution using Playfair but with a novel idea that increases the randomness and nonlinearity of the proposed algorithm. The idea is that, instead of using static coding for converting binary strings to DNA nucleotides, we make the coding random and nonlinear based on the position that is determined by the row corresponding to the first four bits and the column corresponding to the second four bits of the 8-bit word that is being searched for in the 16 × 16 DNA Playfair matrix. Therefore, each one of these bit strings is divided into two four-bit groups; one is used to determine the row, and the other is used to determine its column in the final 16 × 16 DNA Playfair matrix.

The first value, 01111100, is divided into 0111 and 1100, which means that it is located at row 0111 (row number 7) and column 1100 (the column number 12). Note that we are starting from row 0. In the Playfair matrix, it corresponds to TGTG. The second value 01001110 is divided into 0100 and 1110, which means it is located at row 0100 (the row number 4) and column 1110 (the column number 14). In generated Playfair matrix, it corresponds to AGAC. After finding the two four-base DNA words, TGTG and AGAC, we apply rule three of Playfair because they are not in the same row or column, and the two values are replaced by ATGA and GGTT, which are equal to 01111110 and 01001100. We also use the 4-bit row number followed by the 4-bit column number of each word of the substituted words. The encoding is performed by replacing the four DNA nucleotide code with the corresponding 8 bits. The first 4 bits are the row number, and the second 4 bits are the column number.

We go through the red vector and the green vector to perform the same operation with one constraint after encrypting eight bits from the red and green channels. The current key has the XOR performed on it with the sixteen encrypted pixels' values (eight encrypted pixels in the red vector followed by eight encrypted pixels in the green vector), and the new Playfair matrix is generated again. The operation is described in Section 4.4, and it is necessary to defeat differential attacks and to make the whole obtained cipher values different if only one bit is changed in the plain image. The 16 × 16 DNA Playfair matrix after the first XOR with 16 encrypted pixels in the red and green channels is shown in Figure 5.

GGCC	TGCG	GCCT	CAAA	CAAC	CGGG	ACAT	CGTA	CGCC	CGCG	ACCT	TAAA	TCAC	TGGG	TAAT	TGCA
TGTA	GGTC	TGTG	GCTT	CAGA	CAGC	CTAG	ACGT	CTCA	CGTC	CGTG	ACTT	TAGA	TCGC	TTAG	TAGT
TTCA	GTCC	TTCG	GGCT	CCAA	CCAC	CTGG	AGAT	CTTA	CTCC	CTCG	AGCT	TCAA	TGAC	TTGG	TCAT
AAAG	AGGT	AACA	CTTC	CTTG	AGTT	TCGA	TGGC	GAAG	TCGT	TTTA	GTTC	TTTG	GGTT	CCGA	CCGC
AACG	ATCT	TGAA	TTAC	GAGG	TGAT	GACA	TACC	GACG	GTCT	CGAA	CGAC	AAGG	ATAT	AATA	AACC
ACAG	ATGT	ACCA	AATC	AATG	ATTT	TGGA	TTGC	GCAG	TGGT	GATA	TATC	GATG	GTTT	CGGA	CGGC
ACCG	CACT	TTAA	GAAC	GCGG	TTAT	GCCA	TCCC	GCCG	TACT	CTAA	CTAC	ACGG	CAAT	ACTA	ACCC
AGCA	GGAG	ACTG	CATT	TTGA	GAGC	ACTC	TTGT	GCTA	CTGA	GCTG	TATT	TCTC	CTGC	AGAG	CAGT
AGCG	CCCT	GAAA	GCAC	GGGG	GAAT	GGCA	TGCC	GGCG	TCCT	AAAA	AAAC	AGGG	CCAT	AGTA	AGCC
AGTG	CCTT	GAGA	GCGC	GTAG	GAGT	GGTA	TGTC	GGTG	TCTT	AAGA	AAGC	ATAG	CCGT	ATCA	AGTC
ATCG	CGCT	GCAA	GGAC	GTGG	GCAT	GTCA	TTCC	GTCG	TGCT	ACAA	ACAC	ATGG	CGAT	ATTA	ATCC
GCGA	GGGC	TAAG	GCGT	GTTA	TTTC	GTTG	TGTT	ACGA	CACA	CAAG	CGGT	ACGC	ATTC	ATTG	CGTT
CACG	CTCT	GGAA	GTAC	TAGG	GGAT	TACA	GACC	TACG	TTCT	AGAA	AGAC	CAGG	CTAT	CATA	CACC
TCAG	GGGT	TATA	GATC	TATG	TTTT	AGGA	AGGC	CCAG	CTGT	CCCA	CATC	CATG	CTTT	GGGA	GTGC
CCCG	AACT	GTAA	TAAC	TCGG	GTAT	TCCA	GCCC	TCCG	GACT	ATAA	ATAC	CCGG	AAAT	CCTA	CCCC
TCTA	CGAG	TCTG	GATT	ATGA	ATGC	GCTC	AAGT	CGCA	CCTC	CCTG	AATT	GTGA	TAGC	TGAG	GTGT

Figure 5. Modified 16 × 16 DNA Playfair matrix after first XOR with the Key.

4.4. Complementary Rules for Scrambling Blue

Finally, the blue channel is encrypted using one of the complementary rules in Table 3, and the procedure is as follows:

i. After encrypting eight pixels in the red and green vectors, by using the current key, the corresponding eight pixels in the *blue* channel are selected and combined into one vector.
ii. The pixel values are converted into 8-bit binary values, and then these binary values are combined together to form a 64-bit string S.
iii. The string S is divided into 32 pairs of 2 bits.
iv. The current 128-bit key is divided into 32 groups, each of which is 4-bit. Each group is used to scramble one pair of S.
v. Each group is converted into its decimal equivalent D, which is between 0 and 15.
vi. We calculate two values as follows: $rule\_number = D \ mod \ 8$; and $T = D \ mod \ 4$.
vii. Each pair of S is encrypted by choosing the suitable rule from Table 3 according to its *rule_number*, and the rule is applied T times.
viii. After obtaining the 64 encrypted bits, they are merged again into one string and are shifted left by the decimal value of $current\_key[101 : 106]$. Then, they are divided into 8 groups, each of which is 8 bits.
ix. Each group is converted to its decimal equivalent and is stored as an encrypted pixel value in the blue vector.

4.5. Key Mixing

After encrypting 8 pairs of pixels from the red channel and green channel, the current key has the XOR operation performed on it with the 16 obtained encrypted pixels. In more detail, the current key is used to encrypt 8 pairs of pixels in the red and green channels and to choose a suitable complementary rule to encrypt 8 pixels in the blue channel. The encrypted 8 pixels in the *red* channel and the encrypted 8 pixels in the green channel are

arranged into 1 vector. This vector is converted into 128 binary bits and has the XOR operation performed on it with the current key. This guarantees that changes in 1 bit in any channel greatly affects the encrypted image. The final encrypted image and the encrypted three channels are shown in Figure 6.

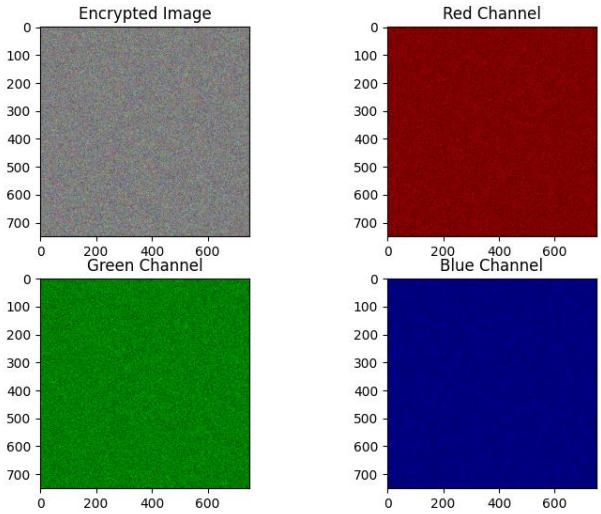

Figure 6. The RGB channels of the final encrypted image.

5. Security Analysis and Evaluation

In this section, a security analysis of the proposed technique is presented and compared with existing techniques. The robustness of any encryption technique is measured by the amount of distortion that it causes to the cipher image. The cipher image should be completely distorted and scrambled. If an attacker obtains a copy of the cipher image, he should still be unable to determine even small numbers of pixels of the original image. In addition to qualitative metrics and visual properties, there are many quantity metrics that are used to evaluate the robustness of encryption techniques. In subsequent subsections, we present each metric along with an analysis and discussion of the proposed technique. These metrics are used to evaluate the robustness of encryption techniques and their ability to resist different kind of attacks, such as cryptanalytic, statistical, brute force and differential attacks. A statistical analysis, key space analysis, sensitivity analysis, resistance to differential attacks and other types of analyses are presented.

5.1. Key Space and Analysis

The basic idea of a brute force attack is that every single possible key is tried until breaking the cipher image. The key should be long enough to make it impossible or practically unfeasible to try all possible keys. On average, half of the possible keys should be tried before the key is found. Once the key is found, all past and future messages that are encrypted with that key are compromised. The overall objective for handling a brute force attack is not to keep data secured forever, but to make the cost of breaking it unfeasible with respect to time consumed and resources. The time consumed to crack a secure message should be very long in such a way that, when the information is found, it becomes useless. Therefore, the only way to resist brute force attacks is to use a large key size to make the algorithm computationally secure. To evaluate the robustness of the proposed technique against brute force and dictionary attacks, a key space analysis and key sensitivity are presented.

5.1.1. Key Space Analysis

The key space should be extremely large to guarantee that it is computationally unfeasible for the brute force attack to find the key and decrypt the cipher image. The total number of possible keys for a cryptographic technique is called the key space. To

resist all kinds of brute force attacks, the key space should be increased dramatically. In our proposed technique, the key length is 128 bits, which implies that there are 2^{128} possible keys. This complies with NIST standards. For example, the original AES algorithm also uses 128 bits key. In the optimistic case that the attacker succeeds in half of the attempts, the attacker requires 2^{127} attempts, or 170141183460469231731687303715884105728 attempts. The attacker finds it impossible to try this key space even with very large distributed computing units.

5.1.2. Key Sensitivity Analysis

Another important measure when evaluating an encryption technique is to evaluate its sensitivity to small changes in the key. If the key changes slightly, the cipher image should be completely changed. Moreover, if one bit is changed in the original key, it should be unable to decrypt the cipher image. In the proposed technique, if a single bit is changed in the key, a completely different cipher image is produced. Moreover, trying to decrypt a cipher image using a key with a single bit modified produces a scrambled image. The percentage of change in the encrypted image and decrypted image using a single-bit modified key exceeds 99%. This indicates that the proposed technique is very sensitive to a slight change in the key; hence, it is able to resist different kinds of brute force attacks. Additionally, the proposed technique depends mainly on the chaotic function, which makes it very sensitive, even to slight changes in initial values.

In this sense, we can guarantee that attackers are not able to restore blocks of pixels, even if they possess a cipher/plain pairs of other pixels. To analyze the amount of change that occurred, a detailed statistical analysis is presented in the next subsection. It is worth noting that the proposed technique is prepared to reflect alterations or modifications to the encrypted image, which means that it can also be used for integrity checks. Therefore, if the encrypted image is cropped or affected by different kinds of noise, it is impossible to recover the original image. However, this happens because the proposed algorithm is pixel-based at the image level. However, our algorithm can also work as block-based by using the initial round key at the beginning of each block, and in this case, all blocks can be restored except the ones that contain modified pixels.

5.2. Statistical Analysis

Statistical analysis is an important metric for evaluating the robustness of a proposed technique against common statistical attacks. A statistical analysis of the encrypted image evaluates the quality of the encryption algorithm with respect to randomness, correlation with the original image and resistance to different kinds of statistical attacks. A statistical analysis of the proposed technique is demonstrated by an entropy analysis and correlation analysis.

5.2.1. Entropy Analysis

Entropy is a statistically random measure that shows how gray pixels are uniformly distributed across an image. It is a common metric that is used to show the randomness of data. An encryption algorithm is evaluated to be robust if the entropy value of the encrypted image approaches eight. For most existing encryption techniques in the literature, the entropy approaches eight, which indicates that the gray scales are uniformly distributed. The value of each pixel is an integer from 0 to 255. If each gray scale occurs in the encrypted image with a probability of 1/256, the entropy of the encrypted image reaches 8 in the ideal case. The entropy [29] is calculated using Equation (2):

$$H(s) = -\sum_{i=0}^{2^N-1} P(s_i) log_2 \frac{1}{P(s_i)} \quad (2)$$

where N is the number of bits used to represent gray scales, and $P(s_i)$ is the probability of occurrence of the variable s_i. In Table 5, the obtained entropy values for standard images are presented. All obtained values are averages of the values obtained when encrypting

the same standard image using 50 different keys. All obtained values are approximately above 7.99, which indicates the robustness and randomness of the proposed technique.

Table 5. Entropy analysis of proposed technique on standard images.

Image	Original			Encrypted		
	R	G	B	R	G	B
Lena	7.2661	7.0122	7.6213	7.9992	7.9952	7.9958
Airplane	7.2384	7.5232	7.1177	7.9894	7.9913	7.992
Baboon	7.7324	7.7545	7.4323	7.996	7.9929	7.9959
Boat	6.9325	7.2164	7.6546	7.9929	7.995	7.9941
Barbara	7.1874	7.0233	7.2233	7.991	7.9888	7.9922
Peppers	7.3423	7.1226	7.5454	7.9944	7.9979	7.9922

Table 6 compares the values obtained for entropy analysis using the proposed technique and existing techniques on the Lena image. The recorded values for the proposed technique are the average values obtained after encrypting the Lena image using 50 different keys. The results show that the proposed technique has competitive comparable results with those that are state-of-the-art. As is clear, all obtained values are above 7.99, which approaches the ideal case. It indicates that the gray scales used in each channel in the encrypted image are evenly distributed. For all tests performed, we carried out an experiment to encrypt the same plain image using 50 different keys and report the averages. We present the averages for metrics to ensure the stability of the proposed technique.

Table 6. Comparison of entropy analysis on Lena image.

Method	R	G	B
Lena	7.2661	7.0122	7.6213
[6]	7.9992	7.9993	7.9994
[17]	7.9993	7.9991	7.9990
[23]	7.9994	7.9993	7.9993
[16]		7.9892	
[24]		7.9959	
[2]		7.9990	
[30]		7.9974	
[24]		7.9959	
[14]		7.9998	
[9]		7.997258	
[5]		7.998	
Proposed	7.9992	7.9952	7.9958

5.2.2. Correlation Analysis

Correlation analysis is the most basic tool used to evaluate the similarity between two images. In cryptography, it is a metric to evaluate the similarity between the encrypted image and the original image. The correlation between adjacent pixels in a plain image and an encrypted image is considered a very important metric for evaluating different encryption techniques. Usually, the correlation between adjacent pixels in a plain image approaches one, because the adjacent pixels are very similar. However, the correlation between adjacent pixels in encrypted images should approach zero to indicate that there

is no relation between adjacent pixels. The correlation analysis [29] is estimated using Equations (3)–(6).

$$C_r = \frac{cov(x,y)}{\sqrt{D(x)D(y)}} \tag{3}$$

where

$$cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))(y_i - E(y)) \tag{4}$$

$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{5}$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2 \tag{6}$$

where N is the total number of pixels selected in our experiments. We randomly pick up 1200 pairs of adjacent pixels (in vertical, horizontal and diagonal directions) from both the original image and the encrypted image. Then, we calculate the correlation between each group. All obtained values for standard images are illustrated in Table 7. Table 8 compares the values obtained for correlation analysis between different techniques when being applied to the Lena image. All reported values for the proposed technique are averages of the values obtained when encrypting the same image using 50 different keys. As is clear, all obtained values are lower than 0.009, which approaches the ideal case.

Table 7. Horizontal, vertical and diagonal correlation analysis of the proposed technique on RGB channels of original and encrypted standard images.

Image		Original Image			Encrypted Image		
		H	**V**	**D**	**H**	**V**	**D**
Lena	R	0.9537	0.942	0.9272	0.002	0.0054	0.0054
	G	0.9811	0.9485	0.9769	0.0037	0.0037	0.0022
	B	0.9574	0.9474	0.9391	−0.0042	−0.0049	0.002
Airplane	R	0.9105	0.9193	0.9324	0.0036	0.0043	0.0028
	G	0.971	0.9701	0.9671	0.0047	0.0054	0.0047
	B	0.9866	0.9212	0.9873	−0.0053	0.0059	0.0019
Baboon	R	0.8919	0.9367	0.8873	0.0035	0.003	0.0051
	G	0.8652	0.8968	0.9224	0.0033	0.0052	0.0023
	B	0.8751	0.9331	0.8808	−0.0057	0.0016	−0.0053
Boat	R	0.9723	0.9619	0.9094	0.0053	0.0037	0.0038
	G	0.957	0.9564	0.904	0.0052	−0.0045	−0.006
	B	0.9009	0.9515	0.9409	0.0032	−0.0054	0.0015
Barbara	R	0.9377	0.9186	0.9387	0.0051	0.0043	0.0043
	G	0.9541	0.9651	0.9256	0.0038	0.0039	0.0019
	B	0.9582	0.9128	0.921	0.0017	−0.0035	−0.0037
Peppers	R	0.9445	0.962	0.9475	0.0044	0.0051	0.0031
	G	0.9714	0.9462	0.9536	0.0031	0.0023	0.0036
	B	0.9763	0.9668	0.9849	0.0049	0.0042	0.0051

Table 8. Horizontal, vertical and diagonal correlation analysis comparison of encrypted Lena image.

Image		Original Images		
		H	**V**	**D**
Lena	R	0.9537	0.942	0.9272
	G	0.9811	0.9485	0.9769
	B	0.9574	0.9474	0.9391
[6]	I	0.0011	−0.0013	−0.0019
[9]	I	−0.025325	0.002558	0.009052
[23]	R	−0.0003	0.0019	−0.0011
	G	−0.0077	−0.0057	100
	B	0.0194	−0.0037	−0.0023
[16]	I	0.014	0.0032	−0.0195
[24]	R	−0.0169	−0.0057	−0.0300
	G	−0.0238	−0.0046	−0.0510
	B	−0.0261	−0.0064	−0.0012
[17]	R	−0.0024	0.0015	−0.0041
	G	−0.0029	−0.0002	0.0004
	B	0.0001	0.0034	−0.0009
[24]	R	−0.0169	−0.0057	−0.0300
	G	−0.0238	−0.0046	−0.0510
	B	−0.0261	−0.0064	−0.0012
[30]	I	0.0044	0.0034	0.008
[15]	I	0.002722	0.001325	0.004133
[14]	I	−0.0003	0.0011	0.0013
[5]	I	−0.00002	0.0008	−0.0002
Proposed	R	0.002	0.0054	0.0054
	G	0.0037	0.0037	0.0022
	B	−0.0042	−0.0049	0.002

5.3. Differential Attack

A differential attack is the most widely used attack in order to analyze the relation between the plain image and the cipher one. In this attack, attackers encrypt the plain image using some key, and then they modify the plain image slightly and encrypt it again with the same key. Then, they compare the two cipher images to try to find the difference between them. The overall objective is to find some relation describing the effect of change in the plain image on the cipher image. Therefore, the attackers always try to make slight changes in the input image and compare the differences between obtained cipher images. This allows the attackers to understand the relationship between the original and encrypted images. The attackers try to determine if a minor change in the gray level of one pixel in the original image has a huge effect on the gray levels of the encrypted image.

To evaluate the effect of a small change in the plain image on the encrypted image, there are two metrics: $NPCR$ and $UACI$. $NPCR$ is the change rate of the number of pixels [12], and it is calculated using Equations (7) and (8).

$$NPCR = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} D(i,j)}{M \times N} \times 100\% \quad (7)$$

$$D(j,i) = \begin{cases} 0 & if\ C_1(i,j) = C_2(i,j) \\ 1 & if\ C_1(i,j) \neq C_2(i,j) \end{cases} \quad (8)$$

where M and N are the number of rows and columns in the cipher images, respectively; C_1 and C_2 are the two obtained cipher images after making a slight change to the plain image and encrypting it using the same key. The UACI is the unified average changing intensity [12], which calculates the mean intensity of the pixel difference in two ciphered images. It is defined by the Equation (9):

$$UACI = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{|C_1(i,j) - C_2(i,j)|}{255} \times 100\% \quad (9)$$

where $C_1(i,j)$ and $C_2(i,j)$ are two corresponding pixels in two cipher images. The cipher images C_1 and C_2 are obtained by encrypting the plain image to produce C_1. Then, one pixel is modified in the plain image and is encryped again using the same key to produce C_2. The values of NPCR and UACI obtained for the standard images are shown in Tables 9 and 10, along with the values for the compared techniques. The reported values for the proposed technique are averages of 50 experiments. In each experiment, we use a different key to encrypt the standard image, and then we modify one pixel within the plain image, encrypt it again with the same key and record the NPCR and UACI.

Table 9. The NPCR values of ciphered images after changing one pixel in their original images.

Image	Technique	NPCR		
		Red Channel	Green Channel	Blue Channel
Lena	[16]		0.995926	
	[23]	0.9960	0.9961	0.9959
	[2]		0.9963	
	[9]		0.996041	
	[14]		0.99606	
	[30]		0.995931	
	[17]	0.9960	0.9964	0.9961
	[5]	0.99615	0.99639	0.99641
	[6]	0.9960	0.9959	0.9964
	Proposed	0.9954	0.9966	0.9961
Baboon	[30]		0.996114	
	[23]	0.996	0.9961	0.9961
	[17]	0.9961	0.9961	0.9960
	[6]	0.9962	0.9962	0.9962
	Proposed	0.99475	0.99685	0.99485
Peppers	[23]	0.996	0.9961	0.996
	[9]		0.996122	
	[30]		0.996221	
	[6]	0.9961	0.9962	0.9962
	[17]	0.9960	0.9962	0.9962
	Proposed	0.9964	0.99405	0.9956
Airplane		0.9956	0.99505	0.9952
Boat		0.994	0.99425	0.9937
Barbara		0.99535	0.9953	0.9943

Table 10. The UACI Values of ciphered images after changing one pixel in their original images.

Image	Technique	UACI		
		Red Channel	**Green Channel**	**Blue Channel**
Lena	[16]		0.3340	
	[23]	0.3308	0.3061	0.2769
	[2]		0.3348	
	[9]		0.334317	
	[14]		0.335126	
	[30]		0.334852	
	[17]	0.3343	0.3348	0.3355
	[5]	0.33477	0.33552	0.33468
	[6]	0.3306	0.3059	0.2760
	Proposed	0.3313	0.33565	0.3343
Baboon	[23]	0.2999	0.2858	0.3123
	[30]		0.332702	
	[17]	0.3348	0.3351	0.3346
	[6]	0.2988	0.2844	0.3104
	Proposed	0.32895	0.33102	0.32805
Peppers	[23]	0.2906	0.3343	0.334
	[9]		0.335431	
	[30]		0.334222	
	[17]	0.3348	0.3346	0.3352
	[6]	0.2892	0.3383	0.3384
	Proposed	0.32675	0.33185	0.32745
Airplane		0.3332	0.332394	0.3293
Boat		0.32885	0.332275	0.32265
Barbara		0.33275	0.33105	0.3281

5.4. Time Analysis

Since our technique is based on the direct manipulation of the bits of each pixel, the complexity of the cryptographic speed is determined by the pixel count in the image, and that makes the proposed technique very efficient with respect to time consumed for encryption and memory allocation. To evaluate the running time of the proposed technique on a core i5 processor with 8GB of RAM, the proposed approach takes 7–9 s on average to encrypt a 256 $\times$ 256 RGB image. However, the proposed algorithm is 16 rounds, which is a large amount of work. A smaller number of rounds can be used, such as 4 rounds of DES, to reduce time constraints, and it is still robust and secure. In the future, we intend to present a parallel version of the algorithm to make it more efficient. Moreover, a customized parallel version of the algorithm can divide the image into blocks, and encrypting these blocks in parallel as if each block was an independent image can reduce the time greatly.

5.5. NIST Analysis

In order to evaluate the randomness of the proposed technique, we examined the proposed technique using the NIST Suite. The NIST tests are published by the National Institute of Standards to test the randomness of encryption techniques. The outcomes of the NIST are listed below, which indicate that the proposed technique passes all random tests and can produce a higher level of randomness, which makes the algorithm resistant

against different attacks. The results for NIST analysis on the encrypted channels of the Lena image are presented in Tables 11–13. As is clear, the encrypted three channels pass all tests. We performed 16 experiments to encrypt the Lena image with different keys, out of which only 2 keys failed on some tests.

Table 11. The NIST analysis of encrypted red channel of Lena image.

Type of Test		***p*-Value**
Frequency Test (Monobit)		0.495
Frequency Test within a Block		0.077
Run Test		0.861
Longest Run of Ones in a Block		0.198
Binary Matrix Rank Test		0.329
Discrete Fourier Transform (Spectral) Test		0.262
Non-Overlapping Template Matching Test		0.814
Overlapping Template Matching Test		0.293
Maurer's Universal Statistical Test		0.751
Linear Complexity Test		0.351
Serial Test:		0.601 0.649
Approximate Entropy Test		0.852
Cumulative Sums (Forward) Test		0.785
Cumulative Sums (Reverse) Test		0.332
Random Excursions Test:		
State	Chi-Squared	*p*-Value
−4	2.402	0.791
−3	5.472	0.36
−2	4.161	0.526
−1	2.353	0.798
1	9.871	0.078
2	1.882	0.865
3	3.589	0.609
4	2.743	0.739
Random Excursions Variant Test:		
State	COUNTS	*p*-Value
−9.0	1060	0.68
−8.0	1069	0.697
−7.0	1118	0.893
−6.0	1093	0.761
−5.0	1025	0.418
−4.0	1083	0.646
−3.0	1168	0.8
−2.0	1186	0.586
−1.0	1159	0.706
1	1118	0.63
2	1110	0.707
3	1063	0.465
4	1042	0.433
5	1063	0.586
6	1096	0.776
7	1139	0.99
8	1214	0.693
9	1297	0.428

Table 12. The NIST analysis of encrypted green channel of Lena image.

Type of Test		*p*-Value
Frequency Test (Monobit)		0.53
Frequency Test within a Block		0.494
Run Test		0.081
Longest Run of Ones in a Block		0.32
Binary Matrix Rank Test		0.828
Discrete Fourier Transform (Spectral) Test		0.797
Non-Overlapping Template Matching Test		0.719
Overlapping Template Matching Test		0.118
Maurer's Universal Statistical Test		0.023
Linear Complexity Test		0.161
Serial Test:		0.428
		0.551
Approximate Entropy Test		0.022
Cumulative Sums (Forward) Test		0.225
Cumulative Sums (Reverse) Test		0.668
Random Excursions Test:		
State	Chi-Squared	*p*-Value
−4	16.382	0.005
−3	9.35	0.095
−2	13.233	0.021
−1	14.684	0.011
1	6.263	0.281
2	0.823	0.975
3	5.27	0.383
4	5.756	0.33
Random Excursions Variant Test:		
State	COUNTS	*p*-Value
−9.0	34	0.911
−8.0	36	0.952
−7.0	39	0.974
−6.0	36	0.944
−5.0	31	0.788
−4.0	35	0.896
−3.0	49	0.572
−2.0	55	0.26
−1.0	50	0.168
1	35	0.73
2	44	0.691
3	51	0.504
4	67	0.208
5	78	0.126
6	106	0.018
7	128	0.004
8	123	0.011
9	121	0.02

Table 13. The NIST analysis of encrypted blue channel of Lena image.

Type of Test	*p*-Value
Frequency Test (Monobit)	0.495
Frequency Test within a Block	0.101
Run Test	0.735
Longest Run of Ones in a Block	0.646
Binary Matrix Rank Test	0.824
Discrete Fourier Transform (Spectral) Test	0.378
Non-Overlapping Template Matching Test	0.103
Overlapping Template Matching Test	0.777

Table 13. *Cont.*

Maurer's Universal Statistical Test		0.187
Linear Complexity Test		0.579
Serial Test:		0.18
		0.09
Approximate Entropy Test		0.729
Cumulative Sums (Forward) Test		0.274
Cumulative Sums (Reverse) Test		0.81
Random Excursions Test:		
State	Chi-Squared	*p*-Value
−4	3.127	0.68
−3	8.295	0.14
−2	13.566	0.018
−1	13.411	0.019
1	7.234	0.203
2	8.519	0.129
3	7.64	0.177
4	6.118	0.294
Random Excursions Variant Test:		
State	COUNTS	*p*-Value
−9.0	477	0.529
−8.0	479	0.513
−7.0	490	0.541
−6.0	507	0.608
−5.0	521	0.669
−4.0	524	0.652
−3.0	535	0.699
−2.0	537	0.642
−1.0	541	0.493
1	628	0.056
2	687	0.034
3	665	0.178
4	634	0.43
5	667	0.306
6	744	0.106
7	788	0.064
8	778	0.099
9	760	0.156

6. Conclusions

In this paper, a novel technique for RGB image encryption is proposed based on chaotic systems and a nonlinear 16×16 DNA Playfair matrix. The proposed technique uses a logistic function to generate round keys, and then it goes through several rounds of DNA encoding, transpositions and substitutions to encrypt individual RGB image pixels. The basic contribution of the proposed technique is that it generates a large number of DNA matrices that are random and nonlinear in nature; it uses a modified version of Playfair to substitute individual pixels at once; and it defeats all kinds of differential attacks by mixing the key with already encrypted pixels. The proposed technique is evaluated and analyzed against common attacks, such as brute force and statistical and differential attacks. The results show that the proposed technique is promising with respect to robustness and randomness. Finally, NIST analysis is presented to ensure the randomness ability of the proposed technique.

Author Contributions: Conceptualization, A.A.A. and K.A.; methodology, A.A.A. and K.A.; software, D.I.; validation, D.I. and M.A.; formal analysis, D.I. and M.A.; investigation, A.A.A. and K.A.; resources, D.I. and M.A.; data curation, A.A.A. and K.A.; writing—original draft preparation, D.I.; writing—review and editing, A.A.A. and K.A.; visualization, D.I.; supervision, A.A.A.; project

administration, K.A.; funding acquisition, A.A.A., K.A. and D.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Arthi, G.; Thanikaiselvan, V.; Amirtharajan, R. 4D Hyperchaotic map and DNA encoding combined image encryption for secure communication. *Multimedia Tools Appl.* **2022**, *81*, 15859–15878. [CrossRef]
2. Paul, L.; Gracias, C.; Desai, A.; Thanikaiselvan, V.; Shanthini, S.; Rengarajan, A. A novel colour image encryption scheme using dynamic DNA coding, chaotic maps, and SHA-2. *Multimed. Tools Appl.* **2022**. [CrossRef]
3. Xue, X.; Zhou, D.; Zhou, C. New insights into the existing image encryption algorithms based on DNA coding. *PLoS ONE* **2020**, *15*, e0241184. [CrossRef] [PubMed]
4. Zhang, Q.; Guo, L.; Wei, X. Image encryption using DNA addition combining with chaotic maps. *Math. Comput. Model.* **2010**, *52*, 2028–2035. [CrossRef]
5. Kumar, V.; Girdhar, A. A 2D logistic map and Lorenz-Rossler chaotic system based RGB image encryption approach. *Multimedia Tools Appl.* **2020**, *80*, 3749–3773. [CrossRef]
6. Liu, Z.; Wu, C.; Wang, J.; Hu, Y. A Color Image Encryption Using Dynamic DNA and 4-D Memristive Hyper-Chaos. *IEEE Access* **2019**, *7*, 78367–78378. [CrossRef]
7. Chen, J. A DNA-based biomolecular cryptography design. In Proceedings of the 2003 International Symposium on Circuits and Systems, ISCAS'03, Bangkok, Thailand, 25–28 May 2003; Volume 3.
8. Gehani, A.; LaBean, T.; Reif, J. DNA-based cryptography. *Asp. Mol. Comput.* **2003**, *2950*, 167–188.
9. Guesmi, R.; Ben Farah, M.A. A new efficient medical image cipher based on hybrid chaotic map and DNA code. *Multimedia Tools Appl.* **2020**, *80*, 1925–1944. [CrossRef]
10. Wu, J.; Liao, X.; Yang, B. Image encryption using 2D Hénon-Sine map and DNA approach. *Signal Process.* **2018**, *135*, 11–23. [CrossRef]
11. Wang, X.; Su, Y. Image encryption based on compressed sensing and DNA encoding. *Signal Process. Image Commun.* **2021**, *95*, 116246. [CrossRef]
12. Chai, X.; Zhi, X.; Gan, Z.; Zhang, Y.; Chen, Y.; Fu, J. Combining improved genetic algorithm and matrix semi-tensor prod-uct (STP) in color image encryption. *Signal Process.* **2021**, *183*, 108041. [CrossRef]
13. Lu, Q.; Zhu, C.; Deng, X. An Efficient Image Encryption Scheme Based on the LSS Chaotic Map and Single S-Box. *IEEE Access* **2020**, *8*, 25664–25678. [CrossRef]
14. Zhang, X.; Hu, Y. Multiple-image encryption algorithm based on the 3D scrambling model and dynamic DNA coding. *Opt. Laser Technol.* **2021**, *141*, 107073. [CrossRef]
15. Liu, Y.; Zhang, J. A Multidimensional Chaotic Image Encryption Algorithm based on DNA Coding. *Multimed. Tools Appl.* **2020**, *79*, 21579–21601. [CrossRef]
16. Cui, G.; Liu, Y.; Zhang, X.; Zhou, Z. A new image encryption algorithm based on DNA dynamic encoding and hy-per-chaotic system, Singapore. In Proceedings of the International Conference on Bio-Inspired Computing: Theories and Applications, Harbin, China, 1–3 December 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 286–303.
17. Wang, X.; Su, Y.; Liu, L.; Zhang, H.; Di, S. Color image encryption algorithm based on Fisher-Yates scrambling and DNA subsequence operation. *Vis. Comput.* **2021**, 1–16. [CrossRef]
18. Huang, Z.-W.; Zhou, N.-R. Image encryption scheme based on discrete cosine Stockwell transform and DNA-level mod-ulus diffusion. *Opt. Laser Technol.* **2022**, *149*, 107879. [CrossRef]
19. Babae, M. A novel text and image encryption method based on chaos theory and DNA computing. *Nat. Comput.* **2013**, *107*, 101–107. [CrossRef]
20. Ahmed, K.; El-Henawy, I. Increasing robustness of data encryption standard by integrating DNA cryptography. *Int. J. Comput. Appl.* **2017**, *39*, 91–105. [CrossRef]
21. Liu, H.; Wang, X.; Kadir, A. Image encryption using DNA complementary rule and chaotic maps. *Appl. Soft Comput.* **2012**, *12*, 1457–1466. [CrossRef]
22. Sridevi, A.; Sivaraman, R.; Balasubramaniam, V.; Siva, J.; Thanikaiselvan, V.; Rengarajan, A. On Chaos based duo confusion duo diffusion for colour images. *Multimedia Tools Appl.* **2022**, *81*, 16987–17014. [CrossRef]
23. Liu, L.; Zhang, L.; Jiang, D.; Guan, Y.; Zhang, Z. A simultaneous scrambling and diffusion color image encryption algo-rithm based on hopfield chaotic neural network. *IEEE Access* **2019**, *7*, 185796–185810. [CrossRef]
24. Joshi, A.B.; Kumar, D.; Gaffar, A.; Mishra, D. Triple color image encryption based on 2D multiple parameter fractional discrete Fourier transform and 3D Arnold transform. *Opt. Lasers Eng.* **2020**, *133*, 106139. [CrossRef]
25. Tavazoei, M.S.; Haeri, M. Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms. *Appl. Math. Comput.* **2007**, *187*, 1076–1085. [CrossRef]

26. Singh, N.; Sinha, A. Chaos-based secure communication system using logistic map. *Opt. Lasers Eng.* **2010**, *48*, 398–404. [CrossRef]
27. Kaur, M.; Kumar, V. A Comprehensive Review on Image Encryption Techniques. *Arch. Comput. Methods Eng.* **2018**, *27*, 15–43. [CrossRef]
28. Srivastava, S.S.; Gupta, N. A Novel Approach to Security using Extended Playfair Cipher. *Int. J. Comput. Appl.* **2011**, *20*, 39–43. [CrossRef]
29. Enayatifar, R.; Guimarães, F.; Siarry, P. Index-based permutation-diffusion in multiple-image encryption using DNA se-quence. *Opt. Lasers Eng.* **2019**, *115*, 131–140. [CrossRef]
30. Mahmud, M.; Rahman, A.U.; Lee, M.; Choi, J.-Y. Evolutionary-based image encryption using RNA codons truth table. *Opt. Laser Technol.* **2019**, *121*, 105818. [CrossRef]