IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 February 2023, accepted 14 March 2023, date of publication 23 March 2023, date of current version 28 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3261133

RESEARCH ARTICLE

A Novel Image Cryptosystem Inspired by the Generation of Biological Protein Sequences

MOHAMMAD NASSEF[1,2], MONAGI H. ALKINANI[1], AND AHMED MAHMOUD SHAFIK[3]
[1]Department of Computer Science and Artificial Intelligence, College of Computer Science and Engineering, University of Jeddah, Jeddah 23890, Saudi Arabia
[2]Department of Computer Science, Faculty of Computer and Artificial Intelligence, Cairo University, Giza 12613, Egypt
[3]Department of Computer Engineering, Faculty of Engineering, Cairo University, Giza 12613, Egypt

Corresponding author: Mohammad Nassef (mnassef@uj.edu.sa)

This work was supported by the University of Jeddah, Jeddah, Saudi Arabia, under Grant UJ-21-DR-26.

ABSTRACT Image encryption is essential to keep images secure either in personal devices or over the cloud drives. Various robust image cryptosystems have been implemented during the past decades to keep the encrypted images far from illegal decryption by eavesdroppers. Biological sequences contain high density information that introduce new challenges and opportunities to traditional cryptography. This article presents a novel image cryptosystem, namely *Image-to-Protein* (*I2P*), that is based on successively encoding the image pixels into sequences of symbols analogous biological *DNA* and *Protein* sequences, and then scrambling these sequences into frequent and non-frequent patterns which in turn are binary encoded into the encrypted file. The proposed cryptosystem has been applied on experimental grayscale and color images. The large key space and the complicated *Protein* encoding operations guaranteed resistance to many known cryptographic attacks. Furthermore, the biological encoding of the plain image pixels resulted in high similarity that achieved noticeable lossless compression when compressed using known text compression techniques.

INDEX TERMS Image encryption, image cryptosystem, biological sequences, DNA, DNA computing, protein, security.

I. INTRODUCTION

Image encryption became an essential duty in the era of open communication and cloud storage. A two-dimensional raster image [1] is represented as a matrix of pixels. Each pixel represents a point in the given image. A grayscale image consists of one channel. This means that each pixel in a given grayscale image is represented by one byte (a value between 0 and 255) that holds the intensity of that pixel. The value 0 is given for pure black pixels, whereas the value 255 is assigned for pure white pixels. Alternatively, an RGB color image is composed of three channels: R, G, and B. Every pixel in a given image consists of three bytes: a byte for the red intensity, a byte for the green intensity, and a byte for the blue intensity. Figure 1 shows both grayscale and RGB color versions of the ''peppers.tif'' image.

The associate editor coordinating the review of this manuscript and approving it for publication was Joewono Widjaja.

A. TYPES OF CRYPTOSYSTEMS

A cryptosystem is a system used for data encryption. There are different types of data that can be encrypted including text, image, video, audio …etc. The role of a cryptosystem is to change the content of the given data into a different (encrypted/ciphered) content that is hard to recover (decrypt/decipher) unless having the key information used to change the original content [2, Ch. 1]. The harder the recovery of the original content, the more successful and robust cryptosystem. The key information used by the cryptosystem for the encryption/decryption process is simply called the Key. Based on the key type, cryptosystems are classified into two types: Symmetric and Asymmetric. In Symmetric cryptosystems, such as DES [2, Ch. 3] and AES [2, Ch. 4], the same key is used in encryption and decryption. Alternatively, two different but related keys are used in asymmetric cryptosystems: private key and public key. The private key is privately used by the person who perform encryption, whereas the public

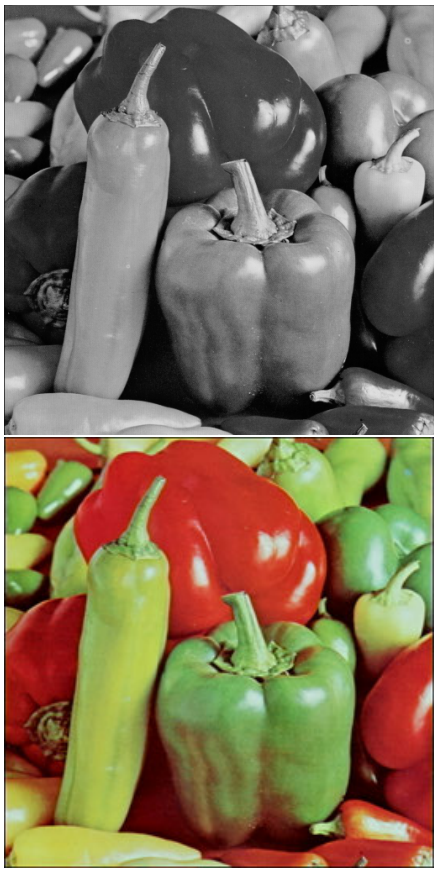

FIGURE 1. Grayscale and RGB color versions of the same "peppers" image. (https://links.uwaterloo.ca/).

key is used by person(s) who can perform the decryption process. Elgamal [2, Ch. 8] and DSA [2, Ch. 10] are examples of known asymmetric cryptosystems.

B. BIOLOGICAL PRELIMINARIES

Almost every cell of the human body contains a complete clone of his/her genetic material. That genetic material is composed of lengthy double-stranded *DNA* sequences (Figure 2). The Central Dogma process [3] describes how parts of these *DNA* sequences (namely Genes) are converted into *mRNA* sequences which in turn are used in generating the different proteins required for the survival of the human body. The invention of sequencing machines three decades ago helped in converting the long biological *DNA* sequences into digital *DNA* sequences [4], [5], [6], [7]. These digital sequences can then be analyzed and manipulated by computer algorithms to explore reasons and possible treatments for various genetic diseases. Furthermore, computer algorithms are used to convert *DNA* sequences into *Protein* sequences [8].

As illustrated in Figure 2, every *DNA* sequence consists of two strands: original and complementary. Each strand is built from 4 different chemical bases that are digitally represented by the four symbols A, C, G, and T. The sequence GTCGGTCCGTCAGTCAGTCCGTCG is an example strand of a digitized *DNA* sequence with 24 symbols. The complementary strand corresponding to the past original *DNA* strand is CAGCCAGGCAGTCAGTCAGGCAGC according to the following mapping: $\{G \rightarrow C, C \rightarrow G, T \rightarrow A, A \rightarrow T\}$. The *RNA* sequence should be derived from the complementary *DNA* strand. However, a computational shortcut can be implemented to directly obtain the *RNA* sequence from the original *DNA* strand by converting every "T" symbol to "U". So, the *RNA* sequence corresponding to the past *DNA* sequence is GUCGGUCCGUCAGUCAGUCCGUCG.

Next, every three symbols in the *RNA* sequence, namely a *Codon*, are used to build one *Protein* unit that is called *AminoAcid*. According to the Central Dogma process, a *Protein* sequence is usually formed from 20 different *AminoAcids* as follows: A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, and Y. Dividing the past *RNA* sequence into multiple *Codons* results in the following sequence of *Codons*: GUC-GGU-CCG-UCA-GUC-AGU-CCG-UCG. According to Figure 3, the *AminoAcids* corresponding to the past sequence of *RNA Codons* are V-G-P-S-V-S-P-S. So, the final *Protein* sequence corresponding to the past *RNA* sequence is VGPSVSPS. It is worthy to note that unneeded parts of the *RNA* sequence, namely Introns, are cut in a process called Splicing before using the remaining parts (Exons) in generating the *Protein* sequence.

A *k-mer* is a subsequence of length k from a given sequence. For the above *Protein* sequence VGPSVSPS, if k equals 3, then the *k-mers* of length 3 (called 3-mers) are as follows: VGP, GPS, PSV, SVS, VSP and SPS. The frequency of a given *k-mer* is its number of occurrences in a given sequence. All the previous *k-mers* have the same frequency (1) because non of them is repeated in the given sequence.

A few decades ago, biologists discovered that the biological systems are working perfectly inside live beings. The proposed *Image-to-Protein* (*I2P*) cryptosystem encrypts a given image by converting its channel(s) into long *Protein* sequence(s) inspired by the continuous generation of biological *Protein* sequence(s) from its *DNA* sequence(s). The shuffling and permutation of image pixels into the space of textual *Protein* sequence(s) achieves the target complexity for a robust cryptosystem.

The coming sections of this article are organized as follows. Section II summarizes the related research efforts, whereas Section III details the proposed *I2P* cryptosystem. The experimental results are listed and discussed in Section IV. The last section concludes the article.

II. RELATED WORK

This section highlights some of the published research efforts that used chaotic maps [9], [10] and biological sequences, namely *DNA* sequences, as a part of their proposed image cryptosystems. The ideas proposed in the following research articles share major parts in common. All these articles depend on the scrambling capabilities provided by miscellaneous chaotic maps. In addition, most of these articles depend on the scrambling capabilities provided by the *DNA* algebraic operations that are performed on *DNA* sequences encoded

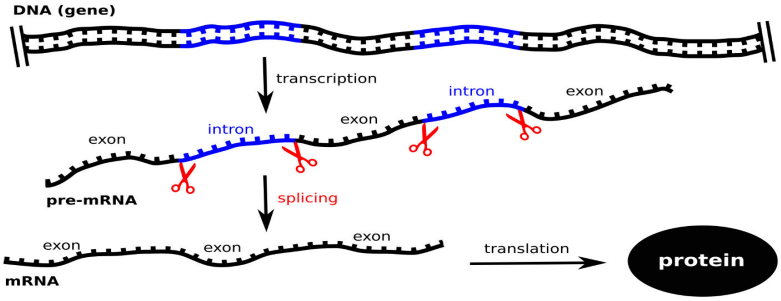

FIGURE 2. The Central Dogma process in Biology that explains how *Protein* sequences are generated from *DNA* sequences. (www.helmholtz-muenchen.de).

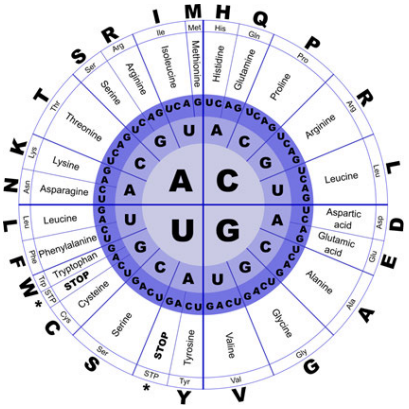

FIGURE 3. The mapping of *RNA Codons* into the corresponding *Protein AminoAcids* [8, p.185].

form the input image pixels. However, each article had its own fingerprint.

Various research articles used different chaotic maps in building different image cryptosystems [11], [12], [13]. The cryptosystem introduced in [11] was based on both true random numbers and chaotic maps, whereas the cryptosystem proposed in [12] depended on both chaotic maps and deep learning to scramble the input image. Alternatively, chaotic signals with finite-precision error have been used by the cryptosystem proposed in [13].

Alternatively, some research efforts made use of biological DNA sequences and operations in image encryption. Zhang et al. [14] and Soni et al. [15] proposed similar image cryptosystems that are based on encoding image pixels into a *DNA* matrix and then adding multiple *DNA* subblocks of that matrix according to a chaotic sequence. Although the experimental results showed that the proposed system is resistant to miscellaneous security attacks, the reversal of the used sorting operation is ambiguous. However, the DES encryption algorithm is used in [15] to encrypt the final image. Moreover, they worked on grayscale images only.

Maniyath and Supriya [16] introduced another image cryptosystem that is fed by *DNA* sequences as input keys. Beside scrambling the input image using the Arnold chaotic maps [17], the input *DNA* sequences were successively XOR-ed with subsequent versions of the image to obtain the final encrypted image. The introduced system was applied to experimental videos. However, the robustness of the encrypted image was mainly based on the size and nature of the image in addition to the number of scrambling and XORing iterations. A similar idea but with different chaotic maps is applied in [18].

Zhang et al. [19] proposed another image cryptosystem that executes *DNA* operations (such as concatenation, insertion, and deletion) over the *DNA* sequences representing the image pixels based on chaotic sequences generated from logistic maps. The proposed cryptosystem was working on grayscale images only. Moreover, the horizontal correlation of pixels in the encrypted image is high, and the algorithm is not resistant to differential attacks.

Zhang et al. [20] proposed an image encryption algorithm that is based on diffusion and confusion of image pixels. First the image pixels are exchanged using a chaotic map. Next, the diffusion phase is performed by converting each pixel into four *DNA* symbols. The confusion phase is based on iteratively replacing every *DNA* symbol by another *DNA* symbol based on a randomly selected *DNA* complementary rule.

Gupta and Jain [21] introduced an improved algorithm that is twofold. The first phase maps the pixels of the given grayscale image into multiple *DNA* matrices that are added later using *DNA* addition operations. In the second phase, the image pixels are scrambled using multiple 1D and 2D logistic maps. The algorithm works on grayscale images only.

Zhang et al. [22] fed a specific *DNA* sequence from a specific public biological database to the Keccak hash function which in turn generates image scrambling parameters. These parameters are then used by multiple chaotic maps to successively scramble grayscale images on both bitwise and pixel levels. After that, the scrambled image is further encoded using multiple algebraic *DNA* operations. The algorithm is very complex, however, its runtime complexity is not shown.

Norouzi and Mirzakuchaki [23] proposed a threefold grayscale image cryptosystem that is based on *DNA* operations and Cellular Neural Network (CNN). The image is initially divided into four *DNA* matrices that undergone algebraic *DNA* operations. Next, a key stream is generated from the input secret key using CNN. Finally, the image pixels are scrambled using chaotic sequences generated by CNN.

Slimane et al. [24] introduced a cryptosystem that is also based on chaotic maps and *DNA* mapping. They used a Single Neuron Model to generate the key stream used to perform simultaneous *DNA* XOR operation with the *DNA* matrices resulting from the three RGB channels of the plain color image.

In [25], to obtain an encrypted image, Zhang et al. performed *DNA* addition over three *DNA* matrices obtained from a real *DNA* sequence represented as a matrix, an encoded *DNA* matrix of the input grayscale image, and a *DNA* matrix resulting from an encoded chaotic sequence. Finally, a selective transformation step is applied over the resultant *DNA* matrix to obtain the final encrypted image. The encryption algorithm was not mentioned in detail.

Feng et al. [26] claimed that most image cryptosystems merely based on chaotic maps fail to resist chosen-plaintext attacks. Thus, they illustrated their own complicated algorithm that is based on hash values generated from the input image alongside the chaotic sequences generated from the secret key. Both hash values and chaotic sequences were successively used to perform iterative permutations and *DNA* operations over the input image with the help of Discrete Logarithm generators. They showed simulation results over both grayscale and color images.

Ben Farah et al. [27] used FRactional Fourier Transform (FRFT) in addition to *DNA* XOR operations and Lorenz chaotic maps to encrypt color images. First, SHA hash values are generated from the input image to form the secret key and the chaotic parameters. Additionally, the input image is encoded into *DNA* matrix. After that, the *DNA* matrix of the image undergoes alternative iterations of *DNA* XOR-ing and FRFT. The authors suggested optical implementation of the proposed algorithm using the single-lens Lohmann's configuration.

Xuejing and Zihui [28] presented a new color image cryptosystem based on *DNA* encoding and Spatiotemporal chaotic system. First, a hash value of the input image, that serves as a secret key, is fed to the chaotic generator that generates three chaotic sequences. After that, *DNA* random encoding of the input image is done based on the first chaotic sequence, whereas the second chaotic sequence is used for self-adaptive permutations of the encoded *DNA*. Next, multiple *DNA* insertion/deletion operations are applied to the encoded image using the third chaotic sequence.

Chen et al. [29] introduced another color image cryptosystem that is based on a three-neuron fractional-order discrete Hopfield neural network (FODHNN) and *DNA* sequence operations. The FODHNN is used as a pseudo-random chaotic sequence generator whose parameters are obtained from an external key alongside a hash value of the input image. The obtained chaotic sequences are used to perform 3D projection and confusion to determine the *DNA* encoding rules for each pixel in the image.

Liang et al. [30] proposed a new medical image cryptosystem that employs a five-dimensional three-leaf chaotic system and various *DNA* operations to encrypt the input image. The final secret key stream is generated from two hash values of the input image. Next, a chaotic matrix is generated based on that key to perform *DNA* computing over the encoded *DNA* of the input image.

Wang and Li [31] proposed a new complex color image cryptosystem that is based on Multi-Objective Particle Swarm Optimization (MOPSO). Initially, the encryption key is composed of three subkeys obtained from MOPSO, a hash value of the plain image, and shuffling information. Next, MOPSO is iteratively used to determine the best (key, cipher image) pair from the subsequently generated (keys, cipher images) based on image entropy information and correlation coefficient values. Moreover, a logistic map uses the intermediate key in each iteration to apply chaotic *DNA* operations over the intermediate cipher image. At the end, the time complexity of the algorithm seems to be very high.

Alli and Dinesh Peter [32] introduced a chaotic color image cryptosystem with the help of an auto-encoder induced *DNA* sequence. They claim that the auto-encoder generates a permuted image with less time complexity and noise. A secret key is then initialized with a hash value of the input image. After that, the *DNA* encoded pixels are shuffled using the generated chaotic sequences and then scrambled using the *DNA* XOR operation. The major drawback of the proposed system is the time consumed in training the auto-encoder.

Patel and Veeramalai [33] presented a new image cryptosystem based on combined chaotic maps, Halton sequence, five-dimensional (5D) Hyper-Chaotic System and *DNA* encoding. The secret key is formed by XOR-ing two hash values of the input image before and after scrambling it using the HaLT map. The HaLT map combines both Logistic Tent chaotic maps and Halton sequence. After that, *DNA* encoding, pixel permutations, *DNA* diffusion, and *DNA* decoding operations are performed over the scrambled image respectively based on chaotic sequences obtained from the 5D hyper-chaotic map.

Abdelfattah et al. [34] introduced a new medical image cryptosystem for Wireless Body Area Network (WBAN) based on a novel multi-chaotic map and adaptive *DNA* operations. The authors created a novel multi-chaotic map that consists of Henon, Gaussian and Logistic maps (HGL). A hash value of the input image is fed to HGL to form chaotic sequences used in *DNA* encoding of image pixels based on *DNA* rules. The final image is obtained by performing *DNA* decoding of the intermediate image.

As shown from the past literature review, the main contribution of all the aforementioned research articles depends on the scrambling capabilities of miscellaneous chaotic (or logistic) maps that are immensely used by the research community since more than two decades [9], [10]. The use of *DNA* encoding was so elementary that it could be replaced by any alphabetical/mathematical mappings that are not related to the biological *DNA* sequences and operations. Additionally, the output encrypted file explicitly represents a matrix of the encrypted image.

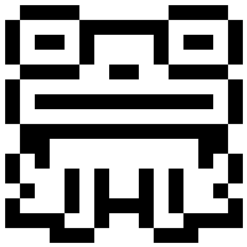

FIGURE 4. Simple frog image with dimensions 16*16 pixels. (http://www.photonstorm.com/art/tutorials-art/16×16-pixel-art-tutorial).

Alternatively, a modest prototype of the *I2P* system [35] was recently published by the authors that extends the biological analogy to *Protein* sequences. Nonetheless, that early *I2P* system could only perform non-secure lossy compression for grayscale images, and it was not mature enough to perform robust lossless image encryption for neither grayscale nor color images. Hence, the current article adds more essential functionalities towards the completeness of the *I2P* framework for robust encryption of both grayscale and RGB color images.

III. THE PROPOSED CRYPTOSYSTEM (*I2P*)

I2P is a symmetric-key cryptosystem. More specific, the same secret key is used for the encryption and decryption processes. As decryption is exactly the reverse process of encryption, this section explains only the encryption algorithm of the proposed cryptosystem. For more clarification of the proposed cryptosystem, all the encryption and decryption steps are applied on the tiny "frog.tif" image depicted in Figure 4. All the intermediate results of that image are attached in the supplementary file entitled "*frog-encryption.pdf*".

A. THE ENCRYPTION ALGORITHM

Figure 5 depicts the complete block diagram for encryption. The encryption algorithm is divided into eight steps starting with the conversion of image pixels into a *DNA* sequence, and ending by encoding the corresponding *Protein* sequence and its related information into a compressed binary file. Furthermore, Table 1 details the fields forming to the initial and final secret keys.

1) SUCCESSIVE TRANSFORMATION OF IMAGE PIXELS INTO *DNA* AND *RNA* SEQUENCES

The encryption process begins with confusing the image pixels into the *DNA* space. More specific, the matrix of pixels should be successively scanned in row-based (or column-based) mode resulting in one dimensional vector of *DNA* patterns. Table 2 shows the mapping of every grayscale pixel into its corresponding *DNA* pattern according to its value. For RGB images, every pixel results in three *DNA* patterns; one pattern for each color byte belonging to that pixel. The generated vector of *DNA* patterns represents the *DNA* sequence of the given image. For example, pixel values 182, 181, 180, 180, 181, 182 result in the *DNA* patterns GTCG, GTCC, GTCA, GTCA, GTCC, GTCG that form the contiguous *DNA* sequence GTCG**GTCC**GTCA**GTCA**GTCC**GTCG**. Next, the *DNA* sequence is converted to the *RNA* sequence: G**U**CGG**U**CCG**U**CAG**U**CAG**U**CCG**U**CG. After that, the successive *Codons* of the *RNA* sequence should be encoded into a sequence of *AminoAcids* forming a *Protein* sequence. As shown in Section I, the direct encoding of the past *RNA* sequence using (Figure 3) results in the *Protein* sequence: VGPSVSPS.

2) SELECTIVE SPLICING OF *RNA* SEQUENCE INTO *PROTEIN* SEQUENCES

The *RNA* sequence produced in the past step is very long and inhomogeneous. The target behind the selective splicing process is to gather the common features of adjacent pixels together into one homogeneous *Protein* subsequence (*OriginalAAs*) and cumulatively retaining the differences among these pixels in another less homogeneous *Protein* subsequence (*CumulativeAAs*).

Intuitively, the flatter a region in an image, the more correlated neighbour pixels in that region. Thus, the difference between the *DNA* patterns representing two neighbour pixels is often in their 4th rightmost character. So, the *RNA* sequence mentioned in the past step can be split by the biological perspective of Exons/Introns Splicing as follows: GUC-**G**-GUC-**C**-GUC-**A**-GUC-**A**-GUC-**C**-GUC-**G**. Traditionally, the successive *Codons* of an *RNA* sequence are directly converted into *AminoAcids* in a *Protein* sequence.

Alternatively, only the left *Codon* (three left characters) of every *RNA* pattern (GUC-**GUC**-GUC-**GUC**-GUC-**GUC**) can be accumulated together resulting in a more homogeneous *AminoAcids* sequence (V**V**V**V**V**V** for the above sequence), namely (*OriginalAAs*). From the biological view, this is analogous to accumulating the Exons that form the final *RNA* strand. After that, the second part (*CumulativeAAs*) is formed by accumulating the 4^{th} character of every *Codon* in the *RNA* sequence (**G**-**C**-**A**-A-C-G). The past six characters are considered as two *Codons* that result in the *AminoAcids* subsequence (**AT**).

Figure 6 illustrates how the *OriginalAAs* and *CumulativeAAs* subsequences are formed. So, the generated *Protein* sequence would be VVVVVV-AT instead of VGPSVSPS that would result from the traditional *Protein* translation process. The application of the selective splicing idea led to higher text similarity inside the *OriginalAAs* subsequence in addition to more diffusion of the original image pixels. Moreover, as will be discussed in the last step, lossless text compression techniques can be applied over the *OriginalAAs* subsequence to obtain very promising compression results.

3) CALCULATING *CodonBits* (*CBs*)

For a cryptosystem to be lossless, the decryption of any encrypted data should result in the exact original data. Therefore, the encrypted data should encode all the information needed to perform lossless decryption, except the key. Hence,

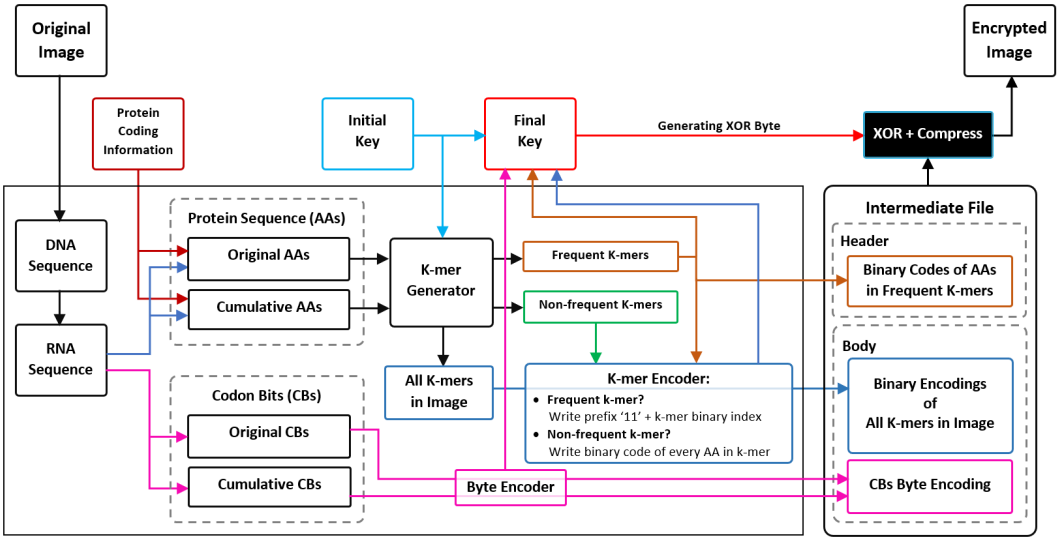

FIGURE 5. Block diagram of the *I2P* Cryptosystem. The initial secret key (Table 1) determines the multiple system parameters for the *k-mers* to be generated. The actual output of the system is two files; one file holds the encoded *k-mers* of the *OriginalAAs* sequence followed by its encoded *OriginalCBs*. The other file contains the encoded *k-mers* of the *CumulativeAAs* sequence followed by its encoded *CumulativeCBs*. The two files are then XOR-ed using information from the final secret key (Table 1) and then compressed together into one file.

TABLE 1. The first seven fields form the initial secret key, whereas the final secret key consists of the overall thirteen fields. The last six fields are determined according to the lengths of bitstreams stored in the intermediate encrypted files.

Field Abbreviation	Field Size in Bytes	Field Sample Value	Field Description
IW	2	512	**I**mage **W**idth
IL	2	512	**I**mage **L**ength
CNL	1	3	**Ch**a**N**ne**L**s (1 for grayscale, 3 for color)
KLO	1	5	***K**-mer* **L**ength for ***O**riginalAAs* sequence
KLC	1	3	***K**-mer* **L**ength for ***C**umulativeAAs* sequence
FKO	2	64	Count of **F**requent ***K**-mers* in ***O**riginalAAs* Sequence
FKC	1	8	Count of **F**requent ***K**-mers* in ***C**umulativeAAs* Sequence (less similarity)
HLO	2	1600	**H**eader **L**ength of Intermediate File for ***O**riginalAAs* sequence (in bits)
LOK	4	1,156,455	**L**ength of Binary Encodings of ***O**riginalAAs* Image ***K**-mers* (in bits)
LOC	3	456,262	**L**ength of Byte Encodings of ***O**riginal**C**Bs* Sequence (in bits)
HLC	2	120	**H**eader **L**ength of Intermediate File for ***C**umulativeAAs* sequence (in bits)
LCK	4	435,680	**L**ength of Binary Encodings of ***C**umulativeAAs* Image ***K**-mers* (in bits)
LCC	3	136,419	**L**ength of Byte Encodings of ***C**umulative**C**Bs* Sequence (in bits)
Total Key Length	28 Bytes 224 Bits		

more binary bits (namely *CodonBits* (*CBs*)) are used in the encoding process to determine from which *RNA Codon* an *AminoAcid* came, achieving lossless decryption. During decryption, *CodonBits* specify the exact *RNA Codon* used to produce an intended *Protein AminoAcid* character during encryption. Table 3 lists the *CodonBits* attached to the different *Codons* that result in the same *AminoAcid*. *CodonBits* are not needed for *AminoAcids* generated from a single *Codon* such as M and W. Figure 7 shows the *CodonBits* '010101010101' corresponding to the illustrative pixel values depicted in Figure 6. According to the *CBs* in Table 3, the 'V' *AminoAcid* can be obtained from four different *Codons*: GUG, GUA, GUC, and GUU. For instance, the *CBs* associated with the 'V' *AminoAcid* and the *RNA Codon* ' GUC' are '01'. Therefore, when reaching a 'V' *AminoAcid* during decompression, the bits '01' in the *CBs* sequence uniquely refer to the 'GUC' *Codon* that the 'V' *AminoAcid* came from. The encoded *CodonBits* increase the size of the encrypted image, but lossless decryption is guaranteed.

4) *k-mer* GENERATION

In this step, the encryption parameters (KLO, KLC, FKO, and FKC) are read from the initial encryption key (Table 1). First, all the possible *k-mers* from both the *OriginalAAs* and *CumulativeAAs Protein* sequences are generated according to the *k-mer* lengths (KLO and KLC) key parameters respectively. Second, the *k-mers* generated from each sequence are sorted in descending order according to their frequencies. After that, *k-mers* of each sequence are divided into two

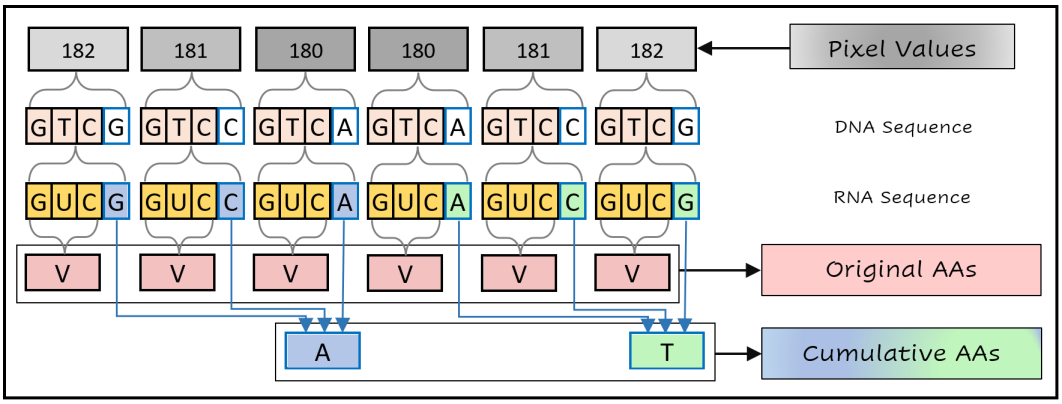

FIGURE 6. An illustrative example of selective translation of six image pixels to the *OriginalAAs* and *CumulativeAAs Protein* sequences. The *DNA Codon* 'GTC' is converted to *RNA Codon* 'GUC' before getting the 'V' *AminoAcid*.

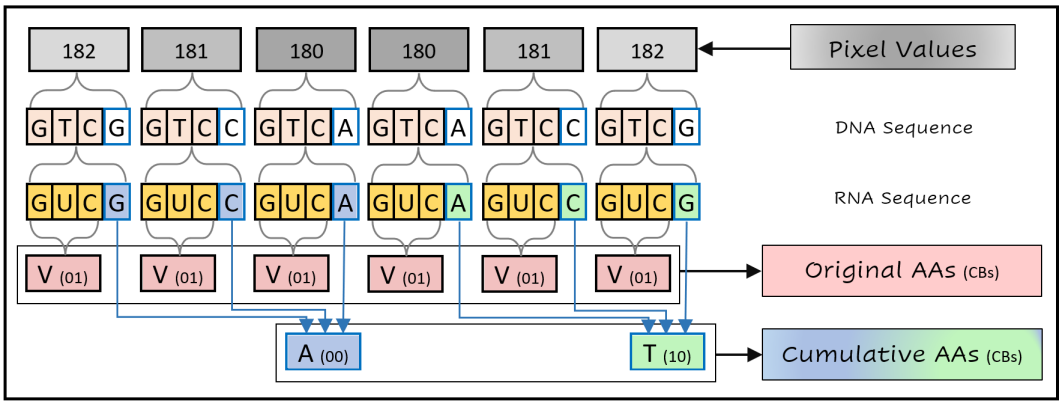

FIGURE 7. The *CodonBits* (*CBs*) generated for the example *OriginalAAs* of the illustrative pixels in Figure 6. According to the *CBs* in Table 3, the 'V' *AminoAcid* can be obtained from four different *Codons*. However, the *CBs* associated with the *RNA Codon* 'GUC' and the 'V' *AminoAcid* are '01'. So, when reaching a 'V' *AminoAcid* during decompression, the bits '01' tells the exact *Codon* that the 'V' *AminoAcid* came from.

TABLE 2. *DNA* alphabet generating 256 single patterns of *DNA*.. AAAA to TTTT.

Pixel Value		Corresponding
Decimal	Binary	*DNA* Pattern
0	00000000	AAAA
1	00000001	AAAC
2	00000010	AAAG
3	00000011	AAAT
4	00000100	AACA
..	.....	....
108	01101100	CGTA
109	01101101	CGTC
110	01101110	CGTG
111	01101111	CGTT
..	.....	....
252	11111100	TTTA
253	11111101	TTTC
254	11111110	TTTG
255	11111111	TTTT

sublists (frequent and non-frequent) according to the (FKO and FKC) parameters.

5) BINARY ENCODING OF *k-mer AminoAcids*

This *k-mer* encoding process will be frequently used during the next subsections for either frequent or non-frequent *k-mers*. It adds more confusion to the plain image. As the number of used *AminoAcid* symbols is 24, 5-bits are enough to uniquely encode any *AminoAcid* (see Table 4). If a *k-mer* includes four *AminoAcids*, then the successive encodings of all its four *AminoAcids* can be accumulated to represent the entire *k-mer* (20-bits). For example, given *k-mer* "KMER", the encodings of its four *AminoAcids* are K = 01000, M = 00100, E = 00011, and R = 01110 respectively. So, the complete encoding of that *k-mer* is 01000001000001101110.

6) BUILDING THE INTERMEDIATE ENCODING FILES

Both intermediate encoding files consist of two sections: a header and a body. The body should contain encodings of all the successive image *k-mers* (*k-mers* of *OriginalAAs* and *k-mers* of *CumulativeAAs* sequences), followed be all the *CodonBits* of the image (*OriginalCBs* and *CumulativeCBs* bitstreams). To write the image *k-mers* to the intermediate encoding files, the *OriginalAAs* and *CumulativeAAs Protein* sequences of the image are traversed to write the encoding of every *k-mer*. If the *k-mer* is non-frequent, its binary AminAcid encodings will be directly accumulated to the body section. Alternatively, the *AminoAcids* encodings of frequent *k-mers* are initially written to the header. Thus, when

TABLE 3. ***DNA Codons*** **and their corresponding** ***CodonBits*** **for the different** ***AminoAcids*****. The M and W** ***AminoAcids*** **do not need** ***CodonBits*****. Three extra fake characters (J, X, and Z) are used to overcome the need for 3-bit binary codes for the L, R, and S** ***AminoAcids*****: (L→J, R→X, and S→Z).**

Codon	*AminoAcid*	*CodonBits*	*Codon*	*AminoAcid*	*CodonBits*	*Codon*	*AminoAcid*	*CodonBits*	*Codon*	*AminoAcid*	*CodonBits*
GCA	A	00	CAC	H	0	CCA	P	00	UCG	S	10
GCC	A	01	CAU	H	1	CCC	P	01	UCU	S	11
GCG	A	10	AUA	I	00	CCG	P	10	ACA	T	00
GCU	A	11	AUC	I	01	CCU	P	11	ACC	T	01
UGC	C	0	AUU	I	10	CAA	Q	0	ACG	T	10
UGU	C	1	AAA	K	0	CAG	Q	1	ACU	T	11
GAC	D	0	AAG	K	1	AGA	R → X	0	GUA	V	00
GAU	D	1	CUA	L	00	AGG	R → X	1	GUC	V	01
GAA	E	0	CUC	L	01	CGA	R	00	GUG	V	10
GAG	E	1	CUG	L	10	CGC	R	01	GUU	V	11
UUC	F	0	CUU	L	11	CGG	R	10	UGG	W	
UUU	F	1	UUA	L → J	0	CGU	R	11	UAC	Y	0
GGA	G	00	UUG	L → J	1	AGC	S → Z	0	UAU	Y	1
GGC	G	01	AUG	M		AGU	S → Z	1	UAA	*	00
GGG	G	10	AAC	N	0	UCA	S	00	UAG	*	01
GGU	G	11	AAU	N	1	UCC	S	01	UGA	*	10

a frequent *k-mer* is hit, its header index is written to the body instead of its *AminoAcids* encodings. The aforementioned encoding of frequent/non-frequent *k-mers* is useful in reducing the length of the body section in addition to more scrambling of the *Protein* sequences representing the image pixels.

As an example of size reduction, assuming that the hits of *k-mer k* are 3,000, *k*'s length is 6 *AminoAcids*, and 32 frequent *k-mers* to be extracted, the traditional encoding of *k-mer k* requires 5*6*3,000 = 90,000 bits. By applying the idea of frequent *k-mers*, *AminoAcids* of *k-mer k* reserve 5*6 = 30 bits in the header bitstream, and the header index of *k-mer k* is $log_2 32 = 5$ bits. So, the 3,000 hits of *k-mer k* in the given image needs 30+(5*3,000) = 15,030 bits only instead of 90,000 bits. After that, the *k-mer* bitstream is encoded into byte stream to be written to the body section of the intermediate file. Similarly, the image *CBs* are encoded into a long list of byte-encodings that are appended to the body section of the intermediate file.

7) XOR-ING THE INTERMEDIATE ENCODING FILES

This step introduces more scrambling and confusion to the bitstreams stored in the intermediate encoding files. First, the lengths (HLO, LOK, LOC, HLC, LCK, and LCC) of bitstreams in intermediate files are appended to the initial key forming the final secret key (Table 1). Second, an XOR byte is calculated based on the contents of the final secret key. The first two XOR bits, namely $Bits_{(b_1,b_2)}$, are calculated using (Equation 1), where the bits with value '1' in the final key are counted and K is the length of the final key in bits. The third, fourth, and fifth bits are calculated using the fields LOK and LCK respectively (Equation 2). The last three bits of the XOR byte are calculated based on fields LOC and LCC as shown in (Equation 3). The final XOR byte is built using (Equation 4). At the end, every byte of the entire intermediate files is XOR-ed with the calculated XOR byte. That way, the modification of any pixel in the original image would result in a different XOR byte, and so, a completely different encrypted content.

$$Bits_{(b_1,b_2)} = ((\Sigma_{i=0}^{K} (x_i = 1))\ modulus\ 3) + 1 \quad (1)$$

$$Bits_{(b_3,b_4,b_5)} = ((LOK + LCK)\ modulus\ 7) + 1 \quad (2)$$

$$Bits_{(b_6,b_7,b_8)} = ((LOC + LCC)\ modulus\ 7) + 1 \quad (3)$$

$$XORbyte_{(b_1 \ldots b_8)} = Bits_{(b_1,b_2)}.Bits_{(b_3,b_4,b_5)}.Bits_{(b_6,b_7,b_8)} \quad (4)$$

8) COMPRESSION OF THE INTERMEDIATE ENCODING FILES

The XOR-ed intermediate files are compressed into a single file using lossless text compression standards such as GZIP, BZIP2, and LZMA. The expected compression ratio is very promising because of the high internal similarity introduced in the bitstreams of each intermediate file, especially the *OriginalAAs* bitstream. The resulting compressed file is considered the final encrypted file of the input image with extension ".i2p".

IV. RESULTS AND DISCUSSION

This section discusses the experimental results and time complexity of the proposed *I2P* cryptosystem. After that, the security of the proposed cryptosystem is analyzed with respect to famous security attacks.

A. EXPERIMENTAL RESULTS

The proposed algorithm has been tested using multiple grayscale and RGB color images from multiple datasets (Table 5). Figure 8 shows the experimental images, whereas, Table 7 lists the experimental results of the attempted images.

As discussed in the past section, the idea of extracting and encoding frequent *k-mers* introduced more scrambling and compression to the final encrypted image. Table 6 lists the most frequent *k-mers* encoded for image "IMG0023.tif" ordered descendingly according to their frequency.

TABLE 4. Binary encodings of the used 24 *Protein AminoAcids*.

Amino Acid	Binary Code	Amino Acid	Binary Code	Amino Acid	Binary Code
A	00000	K	01000	T	10000
C	00001	L	01001	V	10001
D	00010	M	01010	W	10010
E	00011	N	01011	Y	10011
F	00100	P	01100	*	10100
G	00101	Q	01101	J	10101
H	00110	R	01110	X	10110
I	00111	S	01111	Z	10111

The encryption parameters used for experiments in Table 7 are (KLO = 5, KLC = 3, FKO = 64, and FKC = 8). As illustrated in Table 6, the *CumulativeAAs* protein sequence is less homogeneous compared to the *OriginalAAs* sequence. Thus, parameters (KLC and FKC) of *CumulativeAAs* sequence are usually less than the corresponding parameters (KLO and FKO) of *OriginalAAs* sequence. Many experiments have been done using different parameter values, and the idea of frequent *k-mers* proved its effectiveness.

Using other values for these parameters resulted in different encrypted content, however, it did not result in big difference in the size of the encrypted files. For example, using KLO = 9 resulted in longer *k-mers* with fewer frequencies but with longer binary encoding for each *k-mer*, and vice versa. However, using a suitable *k-mer* length and good amount of frequent *k-mers* introduces enough scrambling of the encrypted file in addition to noticeable reduction in its size.

On the other hand, using smaller number of frequent *k-mers* (FKO and FKC) reduces the header size of the encrypted file, but its body section becomes bigger as there will be more non-frequent *k-mers* to be fully encoded. Fully encoding a *k-mer* means writing the binary encoding of every individual *AminoAcid* in the given *k-mer*. Inversely, using relatively bigger number of frequent *k-mers* increases the header size as more bits are needed to uniquely encode every frequent *k-mer*. However, the body section is reduced as there are fewer non-frequent *k-mers* to be fully encoded. At some point, the selected number of frequent *k-mers* will result in encoding naive *k-mers* that are not frequent. In addition, it will result in a big header and an encoding of each *k-mer* in the body section that is longer than the traditional encodings of its *AminoAcids*.

B. COMPLEXITY ANALYSIS

In this section, the time complexity of the proposed cryptosystem is analyzed. The subsequent encoding of the input image into *DNA*, *RNA*, and *Protein* sequences in addition to the generation of a complete list of image *k-mers* costs $O(m.n)$ where m and n are the image's width and length respectively. Sorting the image *k-mers* costs $O(k.log_2(k))$ where k is the total number of *k-mers* in the image's *Protein* sequences. Moreover, writing the header and *k-mers* bitstreams costs $O(k)$, whereas writing the *CBs* bitstream is already a factor in $O(m.n)$. So, the total approximate runtime complexity is $O(m.n + k.log_2(k))$. The past runtime complexity excludes the final compression runtime of the intermediate files.

C. SECURITY ANALYSIS OF THE PROPOSED CRYPTOSYSTEM

In this section, the applicability of various famous security attacks is discussed. It is worthy to note that the encrypted image resulting from the proposed *I2P* cryptosystem lies in the biological domain. So, it is not an explicit encrypted image that retains its size and structure as a matrix of encrypted pixels in the known graphical domain. More specific, the *I2P* encrypted image is represented as one-dimensional stream of bytes resulting from the binary encoding of subsequent biological sequences that migrated the usual graphical domain. Moreover, the byte stream varies in length according to the encryption parameters of the secret key. Thus, cryptanalysis techniques that work on explicit encrypted images might be logically inapplicable to the proposed cryptosystem. The authors did their best to adapt some of these techniques in order to be applicable.

1) BRUTE-FORCE ATTACK (EXHAUSTIVE SEARCH)

In the proposed algorithm, the attacker would try all possible keys to decrypt the encrypted image. As the proposed length of the key is 224 bits, the attacker has to attempt all the key space (2^{224} possible keys, around $2.7 * 10^{67}$ keys) to reach the right key that fully decrypt the encrypted image. So, depending on the nowadays normal computing power, it needs too many years to reach right decryption key. If the computer processing power is 10^6 MIPS, exploring the key space requires $2.7 * 10^{61}$ seconds (around $8.6 * 10^{53}$ years). Thus, the proposed cryptosystem has a large enough key space to resist all kinds of brute-force attacks.

2) SENSITIVITY TO ALTERED PLAIN IMAGE

The sensitivity of the proposed algorithm depends on how the cipher image differs after changing one pixel in the original image. In other words, if one pixel is changed in the original image, how that change affects the *Number of Pixel Change*

TABLE 5. Grayscale and color image datasets used for testing the proposed cryptosystem.

Dataset Title	Image Count	Images Used	URL
Waterloo Greyscale Set 1	12	Grayscale	http://links.uwaterloo.ca/Repository.html
Waterloo Greyscale Set 2	12	Grayscale	http://links.uwaterloo.ca/Repository.html
Waterloo Color Set	8	Color	http://links.uwaterloo.ca/Repository.html
California	39	Grayscale and Color	http://sipi.usc.edu/database/database.php?volume=misc
Lukas	20	Grayscale	http://www.data-compression.info/files/corpora/lukas_2d_8_tif.zip

TABLE 6. Example of most eight frequent *k-mers* (*5-mers* and *3-mers*) for image "IMG0023.tif" from KODAK dataset.

Most 8 Frequent 5-mers from *OriginalAAs* sequence	Frequency	Most 8 Frequent 3-mers from *CumulativeAAs* sequence	Frequency
RRRRR	13,916	KKK	199
PPPPP	13,671	PAV	49
LLLLL	10,734	VGR	48
AAAAA	5,899	GAR	48
VVVVV	5,517	SGA	47
GGGGG	5,513	PRR	46
SSSSS	5,241	RGV	45
IIIII	3,901	GAG	45

FIGURE 8. Attempted experimental images from different datasets.

Rate (NPCR) in the resulting encrypted images according to the following equation:

$$NPCR = \Sigma_{i,j} \frac{D(i,j)}{M \times N} \times 100\% \quad (5)$$

where $D(i, j)$ represents the difference between the values of the same encrypted pixel in the resulting encrypted images. If the difference value is greater than 0, the output is considered 1, whereas, if the difference is 0, the output is considered 0. M and N respectively represent the number of rows and columns in the original image.

Every experimental image in Table 7 is encrypted twice; the first time without any change in the original image, and the second time by altering the middle pixel in the original image.

As the encrypted file of the proposed algorithm consists of a byte stream saved as a binary file, no explicit two-dimensional pixel values stored. Moreover, the size of the encrypted image is also high sensitive to any change in

TABLE 7. Encryption and Security Analysis results of selected images from datasets in Table 5. The used encryption key is as shown in Table 1. The 4^{th} and 5^{th} Columns show the size of each image before and after encryption. Column 6 lists the calculated NPCR values, whereas columns 7 and 8 shows the Correlation-Coefficient value for each plain image and its encrypted file respectively. The last column shows the Entropy values of the encrypted files.

Dataset	Image	Dimensions	Original Size	Encrypted Size	NPCR %	CCA Original	CCA Encrypted	Entropy
Grayscale Images								
California	7.1.02	512*512	256 KB	150.89 KB	99.56	0.969	-0.0101	7.99
California	boat.512	512*512	256 KB	212.62 KB	99.59	0.991	0.0808	7.98
Waterloo 1	Text	256*256	64 KB	**5.28** KB	98.32	0.977	0.0700	7.97
Waterloo 2	Liberary	464*352	159.5 KB	117.60 KB	99.57	0.942	0.0120	7.98
Medical Grayscale Images								
Lukas	hand	1256*1987	2.38 MB	1.192 MB	99.59	0.986	0.0140	7.99
Lukas	foot	1389*2246	2.975 MB	1.215 MB	99.69	0.950	0.1450	7.99
Lukas	knee	1282*2068	2.528 MB	1.286 MB	99.60	0.966	0.0140	7.98
Lukas	head	1673*1556	2.483 MB	**0.992** MB	99.78	0.950	0.1450	7.99
Color RGB Images								
Waterloo 3	Monarch	768*512	1.125 MB	841.23 KB	99.88	0.966	0.0140	7.98
Kodak	Img0007	768*512	1.125 MB	809.68 KB	99.63	0.941	0.1450	7.99
Kodak	Img0021	768*512	1.125 MB	894.09 KB	99.59	0.938	-0.0140	7.98
Kodak	Img0023	768*512	1.125 MB	832.24 KB	99.60	0.905	-0.1050	7.97

the value of one image pixel. For example, the size of the encrypted file of image "img0023.tif" is 852,215 bytes. After altering one pixel in image "img0023.tif", the size of the encrypted file became 852,153 bytes. Another example, the size of the encrypted file of image "text.tif" is 5,414 bytes. Altering one pixel in image "text.tif" resulted in an encrypted file with size 5,449 bytes.

Considering the "7.1.02.tif" original image, its biological mapping "7.1.02.i2p" is represented as a bytestream with length 154,510 bytes. Alternatively, the length of the "7.1.02.i2p" file resulting after altering one pixel in the original image is 154,607 bytes. The two encrypted files differ in 153,926 bytes from their beginning. So, by applying Equation 5, the NPCR value for image "7.1.02.tif" equals 99.56%. This is considered a very satisfying percentage because it implies high sensitivity to the smallest change in the input image.

3) KEY SENSITIVITY

Measuring the key sensitivity is based on comparing the decrypted images of the same encrypted image before and after making a very small modification to the secret key. However, investigating the key sensitivity is practically not applicable for the proposed cryptosystem. Changing any bit in the key is totally not tolerable by the decryption algorithm because the decryption parameters retained in the key become invalid. Mainly, the decryption algorithm would decode wrong XOR byte because the XOR byte depends on the sum of 1's in key's bits in addition to some parameters in the key. Moreover, XOR-ing the intermediate binary files with wrong XOR byte results in wrong binary encodings of *AminoAcids*. That's because the *AminoAcids* binary encodings reserve only 24 of 32 possible encodings that can be built from 5-bits (See Table 4). As a result, the *AminoAcids* decoder would fail to fetch many *AminoAcids* binary encodings and/or their corresponding CodonBits. Moreover, this would impact the correct decoding of both frequent and nonfrequent *k-mers*. At the end, many issues could occur that prevent the decryption algorithm from building correctly decrypted images.

4) KNOWN-PLAINTEXT ATTACK

In a Known-plaintext attack, the attacker has access to samples of plaintext and their corresponding ciphertext. The attacker has to figure out the secret key in order to break other ciphertexts. For the proposed cryptosystem, given two original images and their corresponding encrypted files, the attacker must know the complete secret key to rerun the XOR operation to investigate the details of the encrypted files. Such details include encoding information of *OriginalCBs/CumulativeCBs* and *OriginalAAs/CumulativeAAs* byte streams. This is not straightforward according to the aforementioned computational time needed for brute-force attack. Moreover, knowing a specific key is useful only for a specific image because the parameters (such as image dimensions) stored in each secret key differ from one image to another.

5) CHOSEN-PLAINTEXT ATTACK

In this kind of attack, the attacker has the capability to choose the original image to be encrypted using the encryption algorithm to obtain its corresponding cipher image. In the worst case, the attacker could reveal the secret key. For the proposed cryptosystem, this kind of attack is inapplicable for two reasons. First, the proposed cryptosystem is a symmetric encryption algorithm, so the attacker does not have a public key (like asymmetric cryptosystems) to encrypt the plain image. Second, the image to be encrypted/decrypted requires the whole encryption parameters to produce a complete secret key for encryption/decryption.

Furthermore, a major part (around 64.3%) of the final secret key is not directly fed to the encryption algorithm by the encryptor. More specific, based on Table 1, the last six fields of the key (which are HLO, LOK, LOC, HLC, LCK, and LCC) represent the various lengths of six bitstreams generated in the heart of the encryption process. The lengths of these bitstreams (144 bits) are cumulatively determined in Step.6 of the encryption process, namely Building the Intermediate Encoding Files. So, it is very hard to know these parameters before encryption/decryption. Moreover, the entire secret key is used to determine the XOR byte that is used in Step.7 to further scramble the bytes of the intermediate binary files.

6) CHIPHERTEXT-ONLY ATTACK (OR KNOWN-CIPHERTEXT ATTACK)

This is the weakest kind of the attack because the adversary is assumed to gain access only to a set of ciphertexts. Modern cryptosystems rarely fail under this kind of attack. The adversary tries to analyze a ciphertext without its secret key or its corresponding plaintext. He relies on discovering certain redundancies in the plaintext or discovering the secret key. The adversary might aim to discover parts of the secret key by which he can discover the binary encodings of the encrypted image (such as encodings of *AminoAcids* and frequent *K-mers*) so he can interpret the binary data to image pixels. According to the proposed structure of the final secret key, it might be easy for the attacker to deduce some key parts like (image width, length, and channel). However, the rest of the key (196 bits) is very exhausting to reach to decrypt the cipher image.

7) CHOSEN-CHIPHERTEXT ATTACK

In this attack, the attacker has the chance to choose specific ciphertexts to be decrypted by the cryptosystem. The attacker tries to analyze every ciphertext with its corresponding plaintext to construct information about the used secret key. In the proposed system, the attacker may guess some fields (such as image dimensions) of the final secret key. However, the final secret key consists of different parts that are not related to each other. This will not give him any clue about other key fields such as the bitstream lengths of the intermediate files. Thus, the proposed scheme withstands well against this kind of attack.

8) REPLAY ATTACK

In this type of attack, the adversary intercepts the encrypted content, modify it, and resends it again to the receiver to misdirect him. To success, the adversary has to know the correct arrangements and structure of the encrypted file and the correct encodings of the secret key fields. As mentioned earlier, the decryption algorithm is not tolerable with any modifications in the secret key or the encrypted file. The decryption algorithm will simply halt telling the receiver that there exist inexistant or incomplete binary encodings that cannot be decoded. So, the only way for this kind of attack to success, the adversary must (1) own the right decryption key, (2) decrypt the intercepted ciphertext, (3) modify the obtained plaintext, (4) re-encrypt the plaintext, (5) resend the ciphertext, and (6) resend the secret key to the receiver. It is impossible for the attacker to know such information and implement all the past steps unless the sender (or receiver) is under tough surveillance. Moreover, sending the key in a secure channel and generating hash values for the secret key and the encrypted file guarantees resistance to such an attack.

9) DIFFERENTIAL ATTACK

This attack is somehow similar the Known-plaintext attack, where the attacker has access to a set of original images and their corresponding cipher images. He tries to detect any non-random differences between cipher images of a specific plain image before and after performing specific modifications to that plain image. For example, the attacker might compare the differences between three cipher images of the same plain image with no change, after altering one pixel, and after altering two pixels respectively. This scenario has been applied on the proposed cryptosystem, and NPCR is used to measure the differences between cipher images after changing the value of one pixel and then changing the value of two pixels in the "7.1.02.tif" image. In all cases the differences in NPCR were less than 0.004% which is a negligible ratio for the adversary to deduce the used decrypted key.

10) INFORMATION ENTROPY

This type of analysis represents the degree of information randomness included in the data bytes of the encrypted image. This is calculated by Equation 6 as follows:

$$H = -\Sigma_{i=0}^{L} p(i)\, log_2\, p(i) \tag{6}$$

where $p(i)$ is probability of occurrence of the i^{th} level. For a random 256-level grayscale image bytes (i.e., L = 256), the maximum value of H is 8. Table 7 shows the calculated entropy values of the sample encrypted images which is high enough compared to other related works.

11) GRAY HISTOGRAM ANALYSIS

In general, image histogram describes how the intensities of images pixels are distributed by plotting the number of pixels at each intensity level. The histogram of any plain image is not uniform because it implies the similar intensities of pixels belonging to the same object or region in the plain image. For example, Figure 10 depicts the histogram for the plane image "7.1.02.tif".

Alternatively, the histogram of cipher images resulting from cryptosystems often has a uniform shape, preventing attackers from performing statistical analysis over the cipher images. However, in the proposed cryptosystem, the encrypted file contains a one-dimensional byte stream, not two-dimensional encrypted image. For example, Figure 11 depicts the histogram of the encrypted file of the plane image "7.1.02.tif" with length 154,510 bytes which is uniformly distributed.

FIGURE 9. Grayscale "7.1.02.tif" plain image.

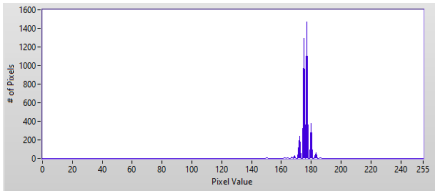

FIGURE 10. Histogram of image "7.1.02.tif".

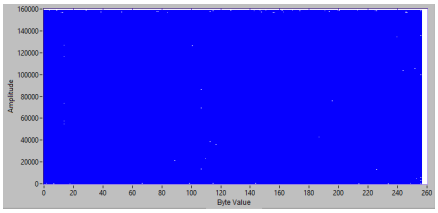

FIGURE 11. Histogram of encrypted image "7.1.02.i2p".

12) CORRELATION COEFFICIENT ANALYSIS (CCA)

In most of the plain images, a high correlation exists between adjacent pixels, whereas there is a little correlation between neighboring pixels in the encrypted image. The following equation is applied to calculate the correlation ratio:

$$C = \frac{N\Sigma_{j=1}^{N}(x_j \times y_j) - \Sigma_{j=1}^{N}(x_j) \times \Sigma_{j=1}^{N}(y_j)}{\sqrt{(N\Sigma_{j=1}^{N}(x_j^2) - (\Sigma_{j=1}^{N}(x_j))^2) \times (N\Sigma_{j=1}^{N}(y_j^2) - (\Sigma_{j=1}^{N}(y_j))^2)}} \quad (7)$$

where x and y are considered the values of two adjacent grayscale pixels, and N represents total number of pixels that are randomly selected to calculate the correlation ratio.

For the proposed cryptosystem, this ratio is measured using bytes extracted from the ".i2p" encrypted files. Table 7 Lists the CCA values of every experimental image as well as its corresponding cipher image. Figures 12 to 17 shows the CCA graphs of images "7.1.02.tif" and "boat.512.tif". CCA is measured using N = 10,000 randomly selected pixels/bytes. In general, the CCA results are considered very good representing a good distribution of the bytes in the encrypted images.

13) DATA LOSS AND NOISE ATTACKS

The Data loss and noise attacks are not applicable to the proposed cryptosystem. First, changing and/or removing bytes from the encrypted file would prevent the successful decompression of the "*.i2p*" file. Assuming that decompression was successful, XOR-ing the intermediate binary files would result in incomplete or noisy bitstreams such that the decryption algorithm will almost fail to decode both frequent and nonfrequent *k-mers*. Moreover, incomplete or noisy bitstreams would include invalid binary encodings that are not assigned to any valid *AminoAcid*. If the binary encodings refer to a valid *AminoAcid*, most probably its corresponding *CodonBits* will be incorrect preventing the successful generation of the image's *DNA* sequence and its pixels.

FIGURE 12. Grayscale "7.1.02.tif" image.

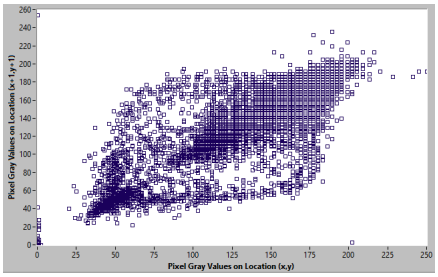

FIGURE 13. CCA of image "7.1.02.tif".

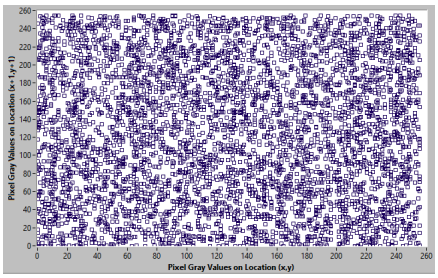

FIGURE 14. CCA of encrypted image "7.1.02.i2p".

D. COMPARATIVE STUDY

In this section, the performance of the proposed cryptosystem is compared against multiple recent research efforts in the literature. It was not easy to perform such comparative study because it is restricted by many parameters such as the availability of cryptosystem implementation, the used runtime environment, and the attempted experimental images with different dimensions and color channels.

Table 8 compares the results collected from different research efforts in the literature with respect to the "lena.tif" grayscale image with dimensions 256 × 256 pixels. The runtime complexity analysis of the proposed cryptosystem is already discussed in Section IV-B. However, the runtime listed in Table 8 was measured based on a Windows 10 PC with Core-i5@2.0GHz processor and 8 GB of memory.

TABLE 8. Comparative study of "lena.tif" encrypted grayscale image with dimensions 256 × 256 pixels.

Algorithm	Image Size (KB) After Encryption	NPCR	CCA Encrypted	Entropy	Runtime (Seconds) Encryption	Runtime (Seconds) Decryption
Ref [20]	64.00	99.7000	0.00280	7.9854	-	-
Ref [22]	64.00	-	0.00820	7.9900	-	-
Ref [23]	64.00	99.6157	-0.00049	7.9980	-	-
Ref [18]	64.00	99.5392	0.01970	7.9975	7.2361	7.4312
Ref [28]	64.00	99.7800	0.00570	7.9993	2.2230	2.3220
Ref [29]	64.00	99.6130	-	7.9975	-	-
Ref [31]	64.00	99.6200	-	7.9975	1.1247	1.1247
Ref [33]	64.00	99.6387	0.00340	7.9920	0.3290	0.2170
Proposed I2P	55.62	99.4700	0.00301	7.9805	0.2791	0.1532

FIGURE 15. Grayscale "boat.512.tif" image.

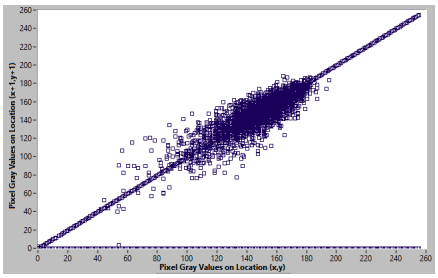

FIGURE 16. CCA of image "boat.512.tif".

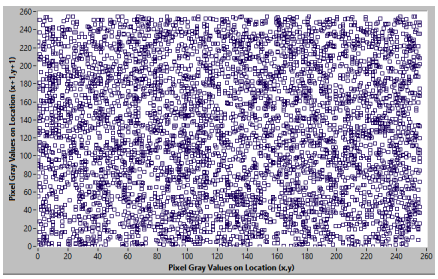

FIGURE 17. CCA of encrypted image "boat.512.i2p".

As shown in Table 8, each algorithm has its own advantages and disadvantages, and no algorithm is perfect among all criteria. The proposed cryptosystem is distinguished by lower runtime and smaller size of the encrypted image in the time it gives acceptable NPCR, CCA and entropy values for the encrypted images.

V. CONCLUSION

The cryptosystem presented in this article suggests a new level of image encryption for both grayscale and colored images. The majority of the past related work encoded image pixels as *DNA* sequences over which algebraic *DNA* operations were applied. The suggested cryptosystem goes deeper to encode image pixels as a confused *protein* sequence that is further diffused into two *protein* subsequences using Alternative Splicing. Moreover, encoding the image's *protein* sequences into frequent and non-frequent *k-mers* added more scrambling and confusion to the image pixels. In addition, XOR-ing the image's bitstreams with information related to bitstreams' lengths guaranteed higher resistance to known security attacks. Furthermore, Alternative Splicing introduced high text similarity to the image's *protein* subsequences resulting in noticeable compression ratios. Although the proposed cryptosystem showed resistance to known security attacks, using dynamic versions of the *DNA/Protein* encoding tables is anticipated to introduce much more resistance to such attacks. Such dynamic versions can be achieved based on parameters from the secret key itself or the content of the input image.

ACKNOWLEDGMENT

The authors would like to thank the University of Jeddah's technical and financial support.

REFERENCES

[1] F. R. Enriquez, *New Basics of Computer Graphics 2020*. Creative Hands Publishing, 2020.
[2] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer, 2010.
[3] P. H. Yockey, *The Central Dogma of Molecular Biology*. Cambridge, U.K.: Cambridge Univ. Press, 2005, pp. 20–26.
[4] J. C. Venter, H. O. Smith, and L. Hood, "A new strategy for genome sequencing," *Nature*, vol. 381, no. 6581, pp. 364–366, May 1996.
[5] J. L. Weber and E. W. Myers, "Human whole-genome shotgun sequencing," *Genome Res.*, vol. 7, no. 5, pp. 401–409, May 1997.
[6] J. C. Venter, M. D. Adams, G. G. Sutton, A. R. Kerlavage, H. O. Smith, and M. Hunkapiller, "Shotgun sequencing of the human genome," *Science*, vol. 280, no. 5369, pp. 1540–1542, Jun. 1998.
[7] G. Myers, "Whole-genome DNA sequencing," *Computing Sci. Eng.*, vol. 1, no. 3, pp. 33–43, May 1999.
[8] P. Compeau and P. Pevzner, *Bioinformatics Algorithms: An Active Learning Approach*, vol. 1. La Jolla, CA, USA: Active Learning Publishers, 2015.
[9] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps," *Int. J. Bifurcation Chaos*, vol. 8, no. 6, pp. 1259–1284, 1998.

[10] M. S. Baptista, "Cryptography with chaos," *Phys. Lett. A*, vol. 240, nos. 1–2, pp. 50–54, Mar. 1998.
[11] S. Zhou, X. Wang, Y. Zhang, B. Ge, M. Wang, and S. Gao, "A novel image encryption cryptosystem based on true random numbers and chaotic systems," *Multimedia Syst.*, vol. 28, pp. 95–112, May 2022.
[12] S. Zhou, Z. Zhao, and X. Wang, "Novel chaotic colour image cryptosystem with deep learning," *Chaos, Solitons Fractals*, vol. 161, Aug. 2022, Art. no. 112380.
[13] S. Zhou, X. Wang, and Y. Zhang, "Novel image encryption scheme based on chaotic signals with finite-precision error," *Inf. Sci.*, vol. 621, pp. 782–798, Apr. 2023.
[14] Q. Zhang, L. Guo, X. Xue, and X. Wei, "An image encryption algorithm based on DNA sequence addition operation," in *Proc. 4th Int. Conf. Bio-Inspired Comput.*, Oct. 2009, pp. 1–5.
[15] R. Soni, A. Johar, and V. Soni, "An encryption and decryption algorithm for image based on DNA," in *Proc. Int. Conf. Commun. Syst. Netw. Technol.*, Apr. 2013, pp. 478–481.
[16] S. R. Maniyath and M. Supriya, "An uncompressed image encryption algorithm based on DNA sequences," *Comput. Sci. Inf. Technol.*, vol. 2, pp. 258–270, Jul. 2011.
[17] G. Chen, Y. Mao, and C. K. Chui, "A symmetric image encryption scheme based on 3D chaotic cat maps," *Chaos, Solitons Fractals*, vol. 21, no. 3, pp. 749–761, Jul. 2004.
[18] T. Hu, Y. Liu, L. H. Gong, and C.-J. Ouyang, "An image encryption scheme combining chaos with cycle operation for DNA sequences," *Nonlinear Dyn.*, vol. 87, no. 1, pp. 51–66, Jan. 2017.
[19] Q. Zhang, X. Xue, and X. Wei, "A novel image encryption algorithm based on DNA subsequence operation," *Sci. World J.*, vol. 2012, pp. 1–10, 2012.
[20] J. Zhang, D. Fang, and H. Ren, "Image encryption algorithm based on DNA encoding and chaotic maps," *Math. Problems Eng.*, vol. 2014, Dec. 2014, Art. no. 917147.
[21] S. Gupta and A. Jain, "Efficient image encryption algorithm using DNA approach," in *Proc. 2nd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2015, pp. 726–731.
[22] X. Zhang, F. Han, and Y. Niu, "Chaotic image encryption algorithm based on bit permutation and dynamic DNA encoding," *Comput. Intell. Neurosci.*, vol. 2017, Aug. 2017, Art. no. 6919675.
[23] B. Norouzi and S. Mirzakuchaki, "An image encryption algorithm based on DNA sequence operations and cellular neural network," *Multimedia Tools Appl.*, vol. 76, no. 11, pp. 13681–13701, Jun. 2017.
[24] N. Ben Slimane, N. Aouf, K. Bouallegue, and M. Machhout, "A novel chaotic image cryptosystem based on DNA sequence operations and single neuron model," *Multimedia Tools Appl.*, vol. 77, no. 23, pp. 30993–31019, Dec. 2018.
[25] T. T. Zhang, S. J. Yan, C. Y. Gu, R. Ren, and K. X. Liao, "Research on image encryption based on DNA sequence and chaos theory," in *Proc. J. Phys., Conf.*, vol. 1004, 2018, Art. no. 012023.
[26] W. Feng, Y.-G. He, H.-M. Li, and C.-L. Li, "A plain-image-related chaotic image encryption algorithm based on DNA sequence operation and discrete logarithm," *IEEE Access*, vol. 7, pp. 181589–181609, 2019.
[27] M. A. B. Farah, R. Guesmi, A. Kachouri, and M. Samet, "A novel chaos based optical image encryption using fractional Fourier transform and DNA sequence operation," *Opt. Laser Technol.*, vol. 121, Jan. 2020, Art. no. 105777.
[28] K. Xuejing and G. Zihui, "A new color image encryption scheme based on DNA encoding and spatiotemporal chaotic system," *Signal Process., Image Commun.*, vol. 80, Feb. 2020, Art. no. 115670.
[29] L.-P. Chen, H. Yin, L.-G. Yuan, A. M. Lopes, J. A. T. Machado, and R.-C. Wu, "A novel color image encryption algorithm based on a fractional-order discrete chaotic neural network and DNA sequence operations," *Frontiers Inf. Technol. Electron. Eng.*, vol. 21, no. 6, pp. 866–879, Jun. 2020.
[30] Z. Liang, Q. Qin, C. Zhou, N. Wang, Y. Xu, and W. Zhou, "Medical image encryption algorithm based on a new five-dimensional three-leaf chaotic system and genetic operation," *PLoS ONE*, vol. 16, no. 11, Nov. 2021, Art. no. e0260014.
[31] X. Wang and Y. Li, "Chaotic image encryption algorithm based on hybrid multi-objective particle swarm optimization and DNA sequence," *Opt. Lasers Eng.*, vol. 137, Feb. 2021, Art. no. 106393.
[32] P. Alli and J. Dinesh Peter, "A novel auto-encoder induced chaos based image encryption framework aiding DNA computing sequence," *J. Intell. Fuzzy Syst.*, vol. 41, no. 1, pp. 181–198, Aug. 2021.
[33] S. Patel and T. Veeramalai, "Image encryption using a spectrally efficient halton logistics tent (HaLT) map and DNA encoding for secured image communication," *Entropy*, vol. 24, no. 6, p. 803, Jun. 2022.
[34] R. Ismail, A. Fattah, H. M. Saqr, and M. E. Nasr, "An efficient medical image encryption scheme for (WBAN) based on adaptive DNA and modern multi chaotic map," *Multimedia Tools Appl.*, pp. 1–15, Jul. 2022. [Online]. Available: https://link.springer.com/article/10.1007/s11042-022-13343-8
[35] M. Nassef and M. H. Alkinani, "A novel multilevel lossy compression algorithm for grayscale images inspired by the synthesization of biological protein sequences," *IEEE Access*, vol. 9, pp. 149657–149680, 2021.

MOHAMMAD NASSEF received the M.Sc. and Ph.D. degrees in computer science from the Faculty of Computer Science and Artificial Intelligence, Cairo University, in 2007 and 2014, respectively. He has been an Associate Professor with the Department of Computer Science, Faculty of Computer and Artificial Intelligence, Cairo University, since 2019. He was an academic supervisor for 15 M.Sc. and Ph.D. students, from 2014 to 2020. He was the Academic Coordinator of the Computing and Bioinformatics Program, Cairo University, from 2016 to 2019. He is currently a Visiting Professor with the Department of Computer Science and Artificial Intelligence, College of Computer Science and Engineering, University of Jeddah, Saudi Arabia. His research interests include machine learning, bioinformatics, genome and image compression, automated essay scoring, and parallel computing.

MONAGI H. ALKINANI received the Ph.D. degree in computer science from Western University, London, Canada, in 2017. In 2018, he joined the Deanship of Scientific Research with the University of Jeddah, Saudi Arabia, where he was the Vice Dean of Research. At the Deanship, he has supervised research in the field of computer vision. Three years later, he was appointed as the Dean of the College of Computer Science and Engineering and within a short period of time, he became the Vice President of Development and Sustainability. He is a member of the Jeddah Computer Vision Team, where he supervises research activities and teaches image processing, artificial intelligence, and signal processing. He has been involved in many collaborative research projects financed by various instances, including the Ministry of Education and the University of Jeddah. In 2022, he was appointed by the Minister of Education as a member of the Alfaisal University's Board of Trustees.

AHMED MAHMOUD SHAFIK received the degree from the Department of Computer Engineering, Faculty of Engineering, Cairo University, Giza, Egypt, in 2017, and the M.Sc. degree in computer security and authentication from Cairo University, in 2019. Since 2019, he has been analyzing and testing many security protocols and platforms. His research interests include computer security, authentication, and cryptography.