Received: December 13, 2022. Revised: February 10, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Unprecedented Security Analysis Results for a Novel Steganography Approach Based on Protein Sequences

Radhwan Jawad Kadhim[1]* **Hussein K. Khafaji[2]**

[1]*Iraqi Commission for Computers and Informatics, Informatics Institute for Postgraduate Studies, Baghdad, Iraq*
[2]*Computer Engineering Department-Al-Rafidain University College, Baghdad, Iraq*
* Corresponding author's Email: phd202110686@iips.icci.edu.iq, hussain.ketan.elce@ruc.edu.iq

Abstract: With the rapid advancement of the digital network, information technology, digital libraries, and especially the services of the world wide web, many types of information can be retrieved at any time. Thus, the issue of security has become one of the most important problems in distributing new information. It is essential to protect this information while it is passing through unsecured channels. Steganography offers a powerful approach to hiding confidential data in suitable media vectors such as images, audio files, text files, and video files. In the field of steganography, the most common measurements of hiding the intended data in a cover media are hiding capacity, cracking probability, payload, and bit per nucleotide (bpn), and these measurements are considered the most of the prime challenges in the field of steganography. The main objective of this research is to provide a novel data hiding schema using protein-based steganography that provides good security performance measurement and outperforms DNA-based steganography approaches, where protein sequences are not utilized for steganography purposes. According to the proposed method, each byte of the secret message is partitioned into two 4-bit parts, and then these parts are converted into the decimal system. Finally, the decimal values of the secret message are randomly included in the cover protein sites using a pseudo random number generator (PRNG) for each cover protein base value instead of being sequentially embedded in the cover protein bases. It is considered that this random distribution increases system security. The proposed method works with a one-digit protein decimal coding rule (PDCR), and the byte of the message will be embedded in two bases of amino acids. From the experimental results, it has been found that the proposed method preserves the reference protein's original function (zero modification rate in the original protein sequence) and achieves a very high hiding capacity of 4 bpn when the sequence's bases are entirely data-embedded, a very low cracking probability, and a payload of zero. Furthermore, the proposed method saves 50% of the bandwidth compared with the other existing technique.

Keywords: Security analysis, Data hiding, steganography, Protein sequence, Protein decimal coding rule, Hiding capacity, BPN, Cracking probability.

1. Introduction

Due to the great advances in information technology and the growing demand for communications, there is an urgent need to transmit all different types of data in a secure communication channel over internet networks to keep sensitive data more secure from unauthorized access. As a result, the transmission of confidential data over a safe communication channel becomes a more challenging problem. Information security safeguards data against illegal access and use, data interruption, disclosure, and modification. Numerous techniques, such as steganography and cryptography, can be used to empower information security [1]. Cryptography is the science that uses a certain algorithm, called an encryption algorithm, to transform the confidential message (plain text) into an incomprehensible format (ciphertext) by using a secret key to change the meaning of the explicit text so that anyone other than the sender and the recipient can't interpret it. The reversing process of cryptography is called decryption [2]. While the science of steganography works on concealing a secret message inside the

cover media (such as images, text, audio, video, and DNA) so that the attacker doesn't know where the secret message is inside the cover media, this is regarded as one of the main benefits of utilizing steganographic techniques [3, 4]. The science of steganography consists of two algorithms: embedding and extracting. The embedding algorithm is used to conceal the confidential data in a cover media to produce a stego-media, which will be sent to the reception side, while the extracting algorithm is used to extract the confidential data from the stego-media. The process of using a key in the embedding and extracting algorithms is optional to increase the level of security [5, 6].

In literature, there are different kinds of steganographic techniques that have been introduced based on the type of carrier medium for the information. The most common methods are text, images, audio, video, and DNA steganography. There are a number of algorithms for image steganography that have been developed, but an image's capacity for hiding information is very low, making it unable to contain a large amount of data [7]. In addition, any distortion of the image leads to attracting the attention of the attacker [8] but the protein sequences are composed of 20 bases (letters), which are meaningless for the majority of people. This means that if the structure of the protein is changed, it does not attract the attention of most people.

Numerous algorithms and approaches have been developed for the steganography techniques mentioned above. However, these strategies have relatively little hiding capacity [7]. In this paper, a novel hiding scheme based on protein sequences is proposed to increase a cover medium's capacity to conceal data. The proposed method increases the hiding capacity to 4 bpn when all bases in the sequence are embedded with data, which outperforms the hiding capacity of all DNA steganographic algorithms. The proposed method for hiding confidential information is considered blind, so it does not need to exchange the original protein sequence. Also, the payload of the proposed approach is zero, and the cracking probability is very low, making it challenging to discern the original message.

The remaining sections of the paper are arranged as follows: A brief biological background for protein synthesis is introduced in section 2. Section 3 presents the motivations and objectives. Section 4 presents briefly related work. The proposed method is described in section 5. Section 6 describes the security analysis. Experimental results are presented and discussed in section 7. Section 8 concludes by summarizing the findings.

Table 1. The genetic code [12]

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

2. Biological background for protein synthesis

To explain the proposed methods of steganography based on bioinformatics concepts, it is necessary to explain the biological background of material synthesis, and the central dogma of molecular biology, is that, how a DNA sequence ultimately led to a protein sequence.

DNA (deoxyribose nucleic acid) is a very large molecule that holds genetic information and features which is very essential for execution and growth of all living organisms [9]. DNA usually consists of two long strands as a double helix running in opposite directions, and each strand is made up of a long chain of sub-units. The sub-units are called nucleotides, where each nucleotide comprises a purine or a pyrimidine base and it is made up of a sugar called deoxyribose, a phosphate group and nitrogenous base. Nitrogenous bases are classified into two classes: the purine bases; adenine (A) and guanine (G), the pyrimidine bases; thymine (T) and cytosine (C)which represent the “genetic code”. Adenine (A) pairs with thymine (T) and guanine (G) pairs with cytosine (C) [10, 11].

The organization of these four bases, which determines the kind of protein molecule and is responsible for all activity in living cells, is crucial to the biological system of living organisms. Additionally, different types of proteins have various functionalities [12]. Through a complicated and long process called central dogma, DNA is transformed into RNA (Ribonucleic Acid), which is considered to be a step in the process of synthesizing proteins, and this process is called transcription [11]. The process of transforming RNA into the amino acids that make up a protein molecule is known as translation. The collection of three consecutive nucleotides, known as a codon, is obtained from RNA during translation,

and each codon designates an amino acid, where the function and structure of the resulting protein are determined by the arrangement of the amino acids [10, 12-14].

Twenty amino acids can be made from different codons, and Table 1 shows that the majority of them can be made from several codons. In addition to amino acids, three STOP codons act as extra indicators for the end of the protein sequence [15].

3. Motivations & objectives

The proposed protein-based steganography is motivated primarily by the desire to improve hiding capacity over DNA-based steganography and other digital media. Due to the complexity of Protein's structure, its computational power as a base-20 numeric system, it's very high levels of redundancy and randomness, and its ability to store huge amounts of data, Protein is used as a cover media for concealing the data, which prompts us to suggest this new method to hide the data. The overall objective of this research is to enhance biological-based steganography methods by considering a high BPN, a very low cracking probability, a high data hiding capacity, a zero payload, a high degree of randomness, preserving a reference protein sequence's functionality, and saving bandwidth. Therefore, the proposed system has very high security against unauthorized access to the data.

4. Related works

Recently, all bioinformatic steganography algorithms use DNA and RNA to conceal data by using their four-base structures as a cover to conceal the data inside. Through our research, we did not find anyone using *a protein sequence* for the purpose of hiding data. In this paper, a protein sequence is used instead of a DNA sequence to conceal the secret message to increase randomness and dispersion, and hence security, because the protein sequence is represented by 20 amino acids. Therefore, in this part, we will discuss the most recent works related to data hiding based on DNA sequence.

Shiu et al. [8] proposed three approaches to concealing data using DNA-based steganography. These approaches are the insertion-based, complementary pair-based, and substitution-based ones. In the insertion-based approach, the secret data and the DNA sequence are converted to binary form and then divided into several segments. At the beginning of each binary segment of the reference DNA sequence, one bit of the secret data is inserted. The size of the DNA file will be expanded along with the size of the data to be hidden in this approach. In the complementary pair-based approach, the secret data is converted to binary form and then partitioned into number of segments in order to select the longest complementary substring pairs. Then, one segment of the secret data is inserted at the beginning of each complementary pair. The original DNA sequence's length is also increased by using this technique. In the substitution-based approach and according to the complementary rules (AC), (CG), (GT) or (TA), each nucleotide base of the reference DNA is substituted with one bit of the secret data. So, in this method, the reference DNA is not expanded, but the cracking probability is high compared with the above two methods. The hiding capacity (bpn) of all three methods is low. Guo et al. [16] proposed a new data hiding approach for DNA-based steganography. The secret data are converted to binary form and then each of two bits are substituted with the repeated character in a DNA sequence according to complementary pairs rules. This method does not work well when the number of secret data segments is less than the number of repeated characters, the BPN is less than one and the cracking probability is high when compared with our method. Taur et al. [17] proposed a data hiding method, where the secret data is concealed in a DNA sequence by substituted each nucleotide base with another nucleotide base based on two bits of the secret data in a lookup table. With this algorithm, the hiding capacity is two secret bits in a nucleotide base, which is still low. Khalifa and Hamad [18] proposed a new hiding method called Least Significant Base Substitution, which exploit codon degeneracy to produce silent mutations into DNA sequences. But this method hides one bit per redundant codon, and does not exploit all the codons for the purpose of hiding, so the data hiding rate is very low. Malathi et al. [19] proposed a data hiding scheme based on DNA steganography. The binary message is XORed with key 1, then the DNA sequence is transformed to binary, segmented into n segments according to key 2, and each bit from the XORed message is inserted at the beginning of each segment to generate a new binary string, which is then converted into a fake DNA sequence. This method suffers from the expansion of the DNA sequence, the cracking probability is high when compared with our proposed method, and the hiding capacity (bpn) is low. Saha et al. [20] proposed a framework for data hiding based on DNA sequences using Balanced Tree Data Structure. In this method, the message is converted to binary form and then to a DNA sequence according to the DNA dictionary rule to get the encrypted DNA message. After that, generate a random cover DNA sequence depending on the size of the encrypted DNA message, then

construct the balance tree with a random DNA sequence. Finally, to obtain the reference DNA sequence, each leaf node in the random DNA sequence is replaced with the DNA message. The size of cover media are increases with the increasing amount of data, Furthermore, the capacity of this method is still low. Mohammed et al. [21] proposed data hiding algorithm based on DNA sequence and neural networks. The binary message is converted to DNA message using the DNA encoding rule. In the reverse order, the resulting DNA message is hidden in the reference DNA and its positions are stored, then convert the positions and the reference DNA into their binary equivalent. Finally, use backpropagation algorithm to train the network by assigning the input with the binary DNA sequence and the target with the binary position. The final weights are transferred to the receiver after accomplishing the training. In this method, the process of training a neural network takes a lot of time to generate final weights, the number of secret bits that can be embedded in the reference DNA sequence in each nucleotide(bpn) for this method is 2 bits which is low, and the cracking probability is high when compared with our proposed method. Mohammed and Abdel-Razeq [22] proposed a method of DNA based steganography using genetic algorithm. The secret message is encrypted using RSA algorithm and converting the result into a binary form, then convert the binary form of the secret message into DNA sequence using the binary coding rule (A = 00; C = 01; G = 10; T = 11). Then, dividing the resultant DNA sequence into segments with key2 (1, 2, 4, 8 etc.). Obtain the best solution of choosing positions in the DNA file Using the genetic algorithm to embed these segments of the secret message. Finally, the fake DNA, key2, and the positions list are sent to the receiver. The number of the secret bits that can be embedded in the reference DNA sequence in each nucleotide(bpn) for this method is 2 bits which is low, the cracking probability is higher than our method, and when the size of the data is very large, the genetic algorithm takes a long time to find the hiding positions. Nabi et al. [23] proposed two new methods based on DNA to hide and encrypt the data, the substitution and insertion methods. In the substitution method, the secret data is concealed in a DNA sequence by substituting each nucleotide base with another nucleotide base based on two bits of the secret data. The BPN of this method is 2, and the cracking probability is high compared with our proposed method. In the insertion method, two DNA sequences (D_A, D_B) are selected from the database based on a private key (K1), then the two sequences must be encoded into binary form according to the second key (K2), and concatenated to produce the binary sequence D_{AB}. In the next step of the algorithm, dividing each binary sequence (D_A and D_B) into segments according to key (K3). After that, the binary message will be divided into two segments; M_1 and M_2. And inserting the bits of segment M_1 at the start of segment D_A, then concatenating these segments to produce D'_A; similarly, inserting the bits of segment M_2 at the start of segment D_B, then concatenating these segments to produce D'_B. According to key (K4), the binary sequence D'_A will be concatenated with the binary sequence D'_B to produce binary sequence D'_{AB}. Then, XORing the binary sequence (D_{AB}) before inserting the message with the binary sequence (D'_{AB}) after inserting the message. Finally, the previous step's binary sequence will be converted back into a DNA sequence using the binary coding rule and sent to the receiver via this step. The drawbacks of this method are: the payload is not zero, the BPN is less than one and the cracking probability is very high compared with our proposed method. Sabry et al. [24] proposed a method for exploiting the redundant codons of amino acids to conceal a secret information in a DNA sequence. The drawbacks of this method are: there is a large expansion in DNA sequence, where the payload is not zero, the BPN is less than one and the cracking probability is high compared with our method. Hassan et al. [25] proposed a hybrid encryption algorithm, where in the first phase, the plain text is encrypted using DNA binary encoding rules and Huffman Coding. Then, the second phase involves hiding a ciphertext into a DNA sequence using least significant base method. The hiding process starts by scanning the DNA sequence from the left to right, and takes the bases positions that are of multiples of 3 (i.e., 3, 6, 9, 12, ... etc), Then, according to some substitution rules and the binary value of the ciphertext (from left to right), the algorithm replaces each base with another base. The main disadvantages of this method are: it requires a DNA sequence that is three times longer than the length of the ciphertext because the number of the secret bits that can be embedded in each nucleotide (bpn) is one bit, and the cracking probability is high compared with our proposed method.

5. Proposed method

To clarify how our proposed method works, we need to explain some important things that related to the process of hiding and retrieving the data:

5.1 Suggested protein decimal coding rule (PDCR)

In bioinformatics, there are twenty amino acids (aa) used to represent the protein molecule, where 5-

Table 2. Protein Decimal Coding Rule (PDCR)

Amino Acid	Three letters Code	One letter Code	5 bits Binary code	One Digit code
Alanine	Ala	A	00000	0
Cysteine	Cys	C	00001	1
Aspartic Acid	Asp	D	00010	2
Glutamic Acid	Glu	E	00011	3
Phenylalanine	Phe	F	00100	4
Glycine	Gly	G	00101	5
Histidine	His	H	00110	6
Isoleucine	Ile	I	00111	7
Lysine	Lys	K	01000	8
Leucine	Leu	L	01001	9
Methionine	Met	M	01010	10
Asparagine	Asn	N	01011	11
Proline	Pro	P	01100	12
Glutamine	Gln	Q	01101	13
Arginine	Arg	R	01110	14
Serine	Ser	S	01111	15
Threonine	Thr	T	10000	16
Valine	Val	V	10001	17
Tryptophan	Trp	W	10010	18
Tyrosine	Tyr	Y	10011	19

bit binary (or one digit in ASCII) is sufficient information to represent the protein molecule. In the proposed method, a one-digit coding rule is created to replace each protein base in the protein sequence with a one-digit code or vice versa, as indicated in Table 2.

Since the amino acids in Table 2 can be arbitrarily assigned, there are totally 20! = (2432902008176640000) possible permutations for this table.

5.2 Suggested amino acid permutation decision table (AAPDT)

As explained in Table 3, the amino acid permutation decision table (AAPDT) is developed to swap out each digit in the decimal protein sequence with one digit that matches the four bits input message. The first row of Table 3 shows that if the amino acid in the reference protein sequence is Glycine (value of 5) and the secret message value is 2 (0010), the resulting fake amino acid is either Leucine (value of 9) or Methionine (value of 10). That is, 9/10 = 𝕣 (5,2). Also, in the reverse process if the amino acid is (Glycine = 5) in the reference protein sequence and the resultant faked amino acid is (either Leucine = 9 or Methionine = 10), then the secret message value is 2 (0010). That is, 2 = 𝕣 (5,9/10). Another case, 1(Cysteine) = 𝕣 (12(Proline), 5(0101)) and 5(0101) =𝕣 (1(Cysteine), 12(Proline)). Keep in mind that the input message can be accurately extracted because the entries (amino acids) in each permutation (*Per*)column must be assigned with different digits. Since the amino acids in Table 3 can be arbitrarily assigned, there are totally $(20!)^{20}$ possible permutations for this table.

5.3 The suggested data hiding algorithm

The Suggested data hiding algorithm is listed in Algorithm 1. To implement this proposed method, it is essential to have four prerequisites in place, which must be known only to the sender and recipient, including a reference protein sequence (*PSI*), protein decimal coding rule (PDCR) and an acid permutation decision table (AAPDT). The proposed algorithm in Step 1 starts by reading the first protein sequence (***PSI).*** These protein sequences can be obtained freely by accessing the NCBI (national center for biotechnology information) database [26]. Step 2 requires the creation of a second fake protein sequence (*PSII*) at random with the same length *PSI*. The algorithm begins with plain text (***M***) and convert it to ASCII code (***MA***), then to binary form (***BM***) as explained in step 3. Step 4 and 5 used to find the random locations as explained in section 5.5. In step 6 and 7, each 8-bits message is divided into two partitions, where each partition is 4-bits partition. After partitioning process, the data must be converted to decimal form ($\boldsymbol{MB'} = \{ m_1', m_2', \ldots, m_{n-1}', m_n' \}$),where one digit in decimal system corresponds to 4 bits in binary system. In order to starts with the data hiding process, the two protein sequences (*PSI* and *PSII*) must be converted to Decimal system according to Table 2 as shown in step 8 and 9. In step 10, let m_j' indicates the secret message value in decimal system, $psi'_{(RL[\,j\,])}$ indicates to an amino acid of a reference protein sequence (in decimal system, see Table 2) in the random location *j*, and$psii'_{(\boldsymbol{RL}[\,j\,])}$ indicates to an amino acid of a faked protein sequence (in decimal system, also see Table 2) with the same random location *j*. Using the Amino Acid Permutation Decision Table, the permutation function $psii'_{(RL[\,j\,])} = \mathbb{r}\,(psi'_{(RL[\,j\,])}, m_j')$ is used to conceal the secret message values. After the permutation process is completed, the decimal values of the amino acids of the two protein sequences are converted to a single letter using Table 2 and send (*PSII*) to the recipient as explained in step 11 and 12 of Algorithm 1.

Table 3. Amino Acid Permutation Decision Table (AAPDT)

0 (Ala)		1 (Cys)		2 (Asp)		3 (Glu)		4 (Phe)		5 (Gly)		6 (His)		7 (Ile)		8 (Lys)		9 (Leu)	
Msg	**Per**	**Msg**	**per**	**Msg**	**Per**	**Msg**	**Per**	**Msg**	**Per**	**Msg**	**Per**	**Msg**	**per**	**Msg**	**per**	**Msg**	**per**	**Msg**	**Per**
0	0	0	1	0	2	0	3	0	4	0	5	0	6	0	7	0	8	0	9
	1		2		3		4		5		6		7		8		9		10
1	2	1	3	1	4	1	5	1	6	1	7	1	8	1	6	1	10	1	11
	3		4		5		6		7		8		9		10		11		12
2	4	2	5	2	6	2	7	2	8	2	9	2	10	2	11	2	12	2	13
	5		6		7		8		9		10		11		12		13		14
3	6	3	7	3	8	3	9	3	10	3	11	3	12	3	13	3	14	3	15
	7		8		9		10		11		12		13		14		15		16
4	8	4	9	4	10	4	11	4	12	4	13	4	14	4	15	4	16	4	17
5	9	5	10	5	11	5	12	5	13	5	14	5	15	5	16	5	17	5	18
6	10	6	11	6	12	6	13	6	14	6	15	6	16	6	17	6	18	6	19
7	11	7	12	7	13	7	14	7	15	7	16	7	17	7	18	7	19	7	0
8	12	8	13	8	14	8	15	8	16	8	17	8	18	8	19	8	0	8	1
9	13	9	14	9	15	9	16	9	17	9	18	9	19	9	0	9	1	9	2
10	14	10	15	10	16	10	17	10	18	10	19	10	0	10	1	10	2	10	3
11	15	11	16	11	17	11	18	11	19	11	0	11	1	11	2	11	3	11	4
12	16	12	17	12	18	12	19	12	0	12	1	12	2	12	3	12	4	12	5
13	17	13	18	13	19	13	0	13	1	13	2	13	3	13	4	13	5	13	6
14	18	14	19	14	0	14	1	14	2	14	3	14	4	14	5	14	6	14	7
15	19	15	0	15	1	15	2	15	3	15	4	15	5	15	6	15	7	15	8

10 (Met)		11 (Asn)		12 (Pro)		13 (Gln)		14 (Arg)		15 (Ser)		16 (Thr)		17 (Val)		18 (Trp)		19 (Tyr)	
Msg	**Per**	**Msg**	**per**	**Msg**	**Per**	**Msg**	**Per**	**Msg**	**per**	**Msg**	**Per**	**Msg**	**per**	**Msg**	**per**	**Msg**	**per**	**Msg**	**Per**
0	10	0	11	0	12	0	13	0	14	0	15	0	16	0	17	0	18	0	19
	11		12		13		14		15		16		17		18		19		0
1	12	1	13	1	14	1	15	1	16	1	17	1	18	1	19	1	0	1	1
	13		14		15		16		17		18		19		0		1		2
2	14	2	15	2	16	2	17	2	18	2	19	2	0	2	1	2	2	2	3
	15		16		17		18		19		0		1		2		3		4
3	16	3	17	3	18	3	19	3	0	3	1	3	2	3	3	3	4	3	5
	17		18		19		0		1		2		3		4		5		6
4	18	4	19	4	0	4	1	4	2	4	3	4	4	4	5	4	6	4	7
5	19	5	0	5	1	5	2	5	3	5	4	5	5	5	6	5	7	5	8
6	0	6	1	6	2	6	3	6	4	6	5	6	6	6	7	6	8	6	9
7	1	7	2	7	3	7	4	7	5	7	6	7	7	7	8	7	9	7	10
8	2	8	3	8	4	8	5	8	6	8	7	8	8	8	9	8	10	8	11
9	3	9	4	9	5	9	6	9	7	9	8	9	9	9	10	9	11	9	12
10	4	10	5	10	6	10	7	10	8	10	9	10	10	10	11	10	12	10	13
11	5	11	6	11	7	11	8	11	9	11	10	11	11	11	12	11	13	11	14
12	6	12	7	12	8	12	9	12	10	12	11	12	12	12	13	12	14	12	15
13	7	13	8	13	9	13	10	13	11	13	12	13	13	13	14	13	15	13	16
14	8	14	9	14	10	14	11	14	12	14	13	14	14	14	15	14	16	14	17
15	9	15	10	15	11	15	12	15	13	15	14	15	15	15	16	15	17	15	18

Algorithm 1. Proposed data embedding algorithm for sender side

Input: A reference Protein sequences (***PSI***), a plain text (***M***), Protein Decimal Coding Rule (***PDCR***), Amino Acid Permutation Decision Table (***AAPDT***)

Output: A faked protein sequences(***PSII***) and ***Key***

Step1. Read a Protein sequence (***PSI***) from NCBI

Step2.Generate a Protein sequence (***PSII***) randomly with same length of ***PSI***

Step3. Covert the plain text (***M***) to ASCII code (***MA***), then binary form (***BM***)

Step4. Find the length of secret message (***ML***) and the length of protein sequence (***PSL)***:

ML(Key) = |***MA***| * 2; PSL = | ***PSI***| or | ***PSII*** |

Step5. Send ***PSL*** and ***ML(Key)*** to Algorithm 3 to find the random locations:

RL = $\{L_1, L_2, \ldots, L_n\}$

Step6. Divide each Byte from (***BM***) in step 3 into two partitions:

MB = $\{m_1, m_2, \ldots, m_{n-1}, m_n\}$ // each partition 4-bit

Step7. Convert each 4-bit partition in step 6 into one digit ASCII code:

$MB' = \{m_1', m_2', \ldots, m_{n-1}', m_n'\}$

Step8. Convert the amino acid in a protein sequence (***PSI)*** into ASCII code as in Table 2:

$PSI' = \mathbb{\Gamma}$ (***PSI)*** = $\{psi_1', psi_2', \ldots, psi_n'\}$

Step9. Convert the amino acid in a protein sequence (***PSII)*** into ASCII code as in Table 2:

$PSII' = \mathbb{\Gamma}$ (***PSII)*** = $\{psii_1', psii_2', \ldots, psii_m'\}$

Step10. For integer ***j*** from **1** to | ***RL*** | // j: is the index of a fragmented message in step 7

$psii'_{(RL[j])} = \mathbb{\Gamma}(psi'_{(RL[j])}, m_j')$//

$\mathbb{\Gamma}$ is a permutation function as in Table 3

Return ***PSII´***

Step11. Convert the digital protein sequence (***PSI´*** and ***PSII´*** **)** from ASCII code into string according to Table 2:

PSI= $\mathbb{\Gamma}(PSI')$

PSII= $\mathbb{\Gamma}(PSII')$

Step12. Send ***PSII*** and ***Key*** to the receiver side

We'll use the example below to demonstrate the steps involved in systematically hiding data: Given a protein sequence ***PSI,*** plain text(***M),*** ***PDCR***, and **AAPDT**

- ***M*** = Stego
- ***PDCR***: the same in Table 2
- **AAPDT**: the same in Table 3

Step1: Read sequence one from NCBI(2NB7_A), ***PSI***= SNAMENTSITIEFSSKF

Step2: Generate sequence two randomly, ***PSII*****=** PHCDKPMKVWMGQNAHE

Step3: Convert a plain text into ASCII, then to binary form:

- **MA = {83, 116, 101, 103,111}**
- **BM =** {01010011, 01110100, 01100101, 01100111, 01101111}

Step4: ***Key*** = |***MA***| * 2 = 10, ***PSL*** = | ***PSI*** | = 17

Step5: Send the values (10, 17) to Algorithm 3:

RL = {3, 9, 10, 13, 5, 15, 11, 16, 14, 8}

Step6: partitioning binary plain text:

- ***MB*** = {0101, 0011, 0111, 0100, 0110, 0101,0110, 0111, 0110, 1111}

Step7: Convert the partitions in step 6 into one digit ASCII code:

- ***MB´*** = {5, 3, 7, 4, 6, 5, 6, 7, 6, 15}

Step 8, Step 9, Step 10 and Step11 are clarified in Table 4.

Step12: Send (***PSII and Key=10)*** to the receiver.

5.4 The proposed data extracting algorithm

In order to extracting the confidential message, the recipient side applies the extraction algorithm as clarified in Algorithm 2. The inputs to the data retrieval algorithm are the same as the inputs to the hiding algorithm in addition to the secret key and the faked protein sequence (*PSII*) containing the hidden data, where the extracting process is the inverse of the hiding process.

Table 4. Example of Hiding a plain text (Stego)

Locations	1	2	**3**	4	**5**	6	7	**8**	**9**	**10**	**11**	12	**13**	**14**	**15**	**16**	17
PSI	S	N	A	M	E	N	T	S	I	T	I	E	F	S	S	K	F
PSII	P	H	C	D	K	P	M	K	V	W	M	G	Q	N	A	H	E
PSII´	12	6	1	2	8	12	10	8	17	18	10	5	13	11	0	6	3
PSI´	15	11	0	10	3	11	16	15	7	16	7	3	4	15	15	8	4
MB´			↓ 5		↓ 6			↓ 15	↓ 3	↓ 7	↓ 6		↓ 4	↓ 6	↓ 5	↓ 7	
PSII´	12	6	9	2	13	12	10	14	14	7	17	5	12	5	4	19	3
PSII	P	H	L	D	Q	P	M	R	R	I	V	G	P	G	F	Y	E
PSI	S	N	A	M	E	N	T	S	I	T	I	E	F	S	S	K	F

Table 5. Example of extracting a plain text (Stego)

Locations	1	2	**3**	4	**5**	6	7	**8**	**9**	**10**	**11**	12	**13**	**14**	**15**	**16**	17
PSI	S	N	A	M	E	N	T	S	I	T	I	E	F	S	S	K	F
PSII	P	H	L	D	Q	P	M	R	R	I	V	G	P	G	F	Y	E
PSI´	15	11	0	10	3	11	16	15	7	16	7	3	4	15	15	8	4
PSII´	12	6	9	2	13	12	10	14	14	7	17	5	12	5	4	19	3

Algorithm 2. Proposed data extracting algorithm for receiver side

Input: A faked Protein sequence (***PSII***), A reference Protein sequence (***PSI***), Protein Decimal Coding Rule (***PDCR***), Amino Acid Permutation Decision Table (***AAPDT***), length of secret message (***Key***).
Output: The hidden plaintext (***M***)
Step1. Find the length of the protein sequence, where ***PSL*** = | ***PSI*** | or | ***PSII*** |
Step2. Send ***Key*** and ***PSL*** to Algorithm 3 to find the random locations: ***RL*** = $\{L_1, L_2, \ldots, L_n\}$
Step3. Convert the amino acid in faked protein sequence (***PSII***) into ASCII as in Table 2:
$\boldsymbol{PSII'} = \mathbb{\Gamma}$ (***PSII***) = $\{psii_1', psii_2', \ldots, psii_m'\}$
Step4. Convert the amino acid in reference protein sequence into ASCII as in Table 2:
$\boldsymbol{PSI'} = \mathbb{\Gamma}$ (***PSI***) = $\{psi_1', psi_2', \ldots, psi_n'\}$
Step5. Initialize a message ($\boldsymbol{MB'}$) as an empty set
Step6. For integer ***j*** from **1** to | ***RL*** |
$\{m_j'\} = \mathbb{\Gamma}\ (\boldsymbol{psii'_{(RL[j])}}, \boldsymbol{psi'_{(RL[j])}})$
$\boldsymbol{MB'} = \boldsymbol{MB'} + \{m_j'\}$
Return $\boldsymbol{MB'}$
Step7. Convert each digit in the set $\boldsymbol{MB'}$ into Binary form ***MB*** to produce 4-bit element
Step8. Concatenate each two elements in step 7:
BM = { $(m_1 + m_2), \ldots, (m_{n-1} + m_n)$ }
Step9. Covert each two concatenated elements in step8 into ASCII code (***MA***)
Step10. Convert ASCII code in step 9 (***MA***) into a plain text (***M***)

To illustrate the process of extracting data step by step, we will take the following example: Given a protein sequence ***PSI***, ***PSII, Key, PDCR***, **AAPDT**

- ***PSI***= SNAMENTSITIEFSSKF
- ***PSII*** = PHLDQPMRRIVGPGFYE
- ***Key*** = 10
- ***PDCR***: the same in Table 2
- **AAPDT**: the same in Table 3

Step1: ***PSL*** = | ***PSI*** | = 17
Step2: Send the values (10, 17) to Algorithm 3:
RL = {3, 9, 10, 13, 5, 15, 11, 16, 14, 8}
Step 3 and Step 4 are clarified in Table 5.
Step5: $\boldsymbol{MB'}$ = { }
Step6: $\boldsymbol{MB'}$ = {5, 3, 7, 4, 6, 5, 6, 7, 6, 15}
Step7: ***MB*** = {0101, 0011, 0111, 0100, 0110, 0101,0110, 0111, 0110, 1111}
Step8: ***BM***= {01010011, 01110100, 01100101, 01100111, 01101111}
Step9: **MA = {83, 116, 101, 103, 111}**
Step10: ***M*** = Stego

5.5 The algorithm of generating the positions of data hiding and extracting using PRNG

To explain how one can find the random positions for hiding or extracting the secret a plain text, the pseudo random number generator (PRNG) was used. A random number generator is utilized to randomly distribute and conceal the bits of a secret plain text into bases within a cover protein sequence. Both the sending and receiving end share the key of stego. The key of stego is considered as a seed to select the number of locations in a cover protein sequence for concealing and extracting the secret plain text and the output are random numbers (L_1, L_2, …, L_n) represents the positions of hiding and extracting a plain text, where n is the required number of locations. So, the size of the key increases with the size of the data. Algorithm 3 starts by searching for the first largest prime number P, such that: $1 \leq P \leq PSL$, then a first primitive root ***a*** is obtained, which is a number whose powers generate all the distinct integers from 1 to (p-1) in a random order [27]. Each power of this primitive root to generate these random integers is called the discrete algorithm. Then this primitive root ***a***, is used to generate a set of random and unrepeated numbers. The number of these random numbers is the same numbers of amino acids in the protein sequence. And according to the value of the key, the number of the required positions is chosen for the purpose of hiding or retrieving the data.

Algorithm 3. Generating data hiding positions using pseudo random number generator

Input: Length of secret message (***Key)*** and the length of protein sequence ***(PSL)***

Output: Random locations ***RL*** = {L_1, L_2, …, L_n}
Step1. Choose the largest prime number ***P,*** such that: $1 \leq P \leq PSL$
Step2. Find the results of first primitive root to ***P***, using:

$$y_i = a^i \bmod p, \qquad a, i = 1, 2, \ldots, P\text{-}1$$

Step3. Find the random locations ***RL***:
For integer ***j*** from 1 to ***Key***
RL [***j***] = ***y*** [j]

6. Security analysis

This section will illustrate some of the most important measurements parameters that is used to evaluate the performance and analysis the security of our proposed algorithm. These measurements are cracking probability, capacity, payload, and BPN (bit per nucleotide) as illustrated below:

6.1 Cracking probability (CP)

A computation of the likelihood that a brute force attempt to guess the secret message concealed in the protein sequence would succeed using the suggested algorithm. To extract the secret message hidden inside the reference protein sequence, the attacker needs to have the following information:

- The number of protein sequences now available online in NCBI database is 1098741385 sequences [26]. Therefore, the probability to predict a reference protein sequence is:

$$\frac{1}{1098741385} = \frac{1}{np}$$

- The probability to predict the protein decimal coding rule (Table 2) is:

$$\frac{1}{20!}$$

- The probability to predict the number of all possible column's permutation (Table 3) are:

$$\frac{1}{(20!)^{20}}$$

- The total probability of guessing each segment of the decimal secret message is: $\frac{1}{2^4} = \frac{1}{16}$, since each decimal segment of a secret message consist of 4-bits.
- The random locations that are used to hide the secret message within a faked protein sequence. The proposed method hides each four secret bits (one digit in decimal) of the binary form of the message (***MB´***) in the faked protein sequence (*PSII*) to conceal it in arbitrary positions. According to the pseudo random numbers generated within [1, |PSII|]. So, the number of required tries to obtain the used set of random numbers is (|PSII|)! So, the cracking probability will be further improved as *PSII* grows. Therefore, the overall likelihood to predict the correct location sets is:

$$\frac{1}{(|PSII|)!}$$

Finally, the total probability to discover the secret message hidden within the reference protein sequence using the proposed method is:

$$C = \frac{1}{np} \times \frac{1}{20!} \times \frac{1}{(20!)^{20}} \times \frac{1}{16} \times \frac{1}{(|PSII|)!} \qquad (1)$$

From Eq. 1, we conclude that the cracking probability of our proposed method is nearly equal to zero.

6.2 Capacity (C)

The capacity refers to the overall length of the extended sequence after the secret message is concealed within it [3], therefore, we can calculate the hiding capacity of the proposed method by the following equation:

$$C = |PSII| \qquad (2)$$

where:
$|PSII|$ is the length of a cover protein sequence.

6.3 Bit per nucleotide (BPN)

Is the average number of secret bits that can be

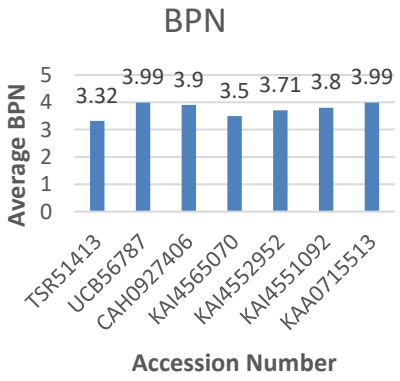

Figure .1 The performance of the proposed algorithm for the same data size on different protein sequences

concealed in each amino acid (character)of the embedded protein sequence. The optimal ***bpn*** of the proposed method is four bits. When all letters in the sequence are utilized to conceal the data, the bpn works as optimal [17].

$$\text{BPN} = \frac{|\mathbf{BM}|}{|C|} \quad (3)$$

where:

$|BM|$ is the total length of a secret message in bits.

6.4 Payload

Is the length of a protein sequence after and before steganography process, and the best value when payload equals zero [22].

$$\text{Payload}=|\text{PSII}|-|\text{PSI}| \quad (4)$$

where:

|PSII| is the length of the fake protein sequence, and |PSI| is the length of the reference protein sequence.

7. Experimental results

7.1 Experimentations on sample protein sequences

This section assesses the effectiveness of the suggested approach. The proposed method was tested on Intel(R) Core (TM) i5-6200U CPU @ 2.4 GHz personal computer with RAM of 4 GB. The MATLAB R2021a program used to implement the proposed method. Seven protein sequences were chosen as test samples, as can be seen in Table 6. Each protein sequence is recognized by its accession number, which is taken from the NCBI database. 22,630 bytes of textual data were randomly selected and used as the secret message. In addition, as shown in Table 6, the capacity provided by the protein sequence, payload, and BPN is shown, as these parameters were described in section 6. From Figure 1, notice that as the sequence size decreases, the BPN increases; in 22,630 bytes of data, if the sequence length was 45260 amino acids, then the BPN was 4.

7.2 Comparison the proposed method with other techniques in term of Capacity

In this sub-section, using different data sizes, the length of the faked DNA sequences is measured by some of the existing techniques to prove the ability of the proposed method compared to the other technique in Table 7. As shown in Table 7 and in the case of (10) KBs, the worst capacity over all methods was achieved by those based on DNA sequences in A4 and A11, which required the highest number of DNA bases (245,760), due to hiding (0.333) bits per DNA base. A2 works by hiding the data in the repeated bases of the DNA sequence, and these bases may not be sufficient to hide the data, so the length of the sequence must be increased for the purpose of searching for the repeated bases and exploiting them in the hiding process. The capacity of A5 is 122,880, bases because of adding extra bases to the original sequence, and these extra bases are the secret message. The capacity of A1 and A10 are 81920 and 87149 bases, due to hiding (1 and 0.94) bits per DNA base, respectively. In A6, the secret message must be hidden in the leaf nodes, and there are 46,656 leaf nodes for hiding 81920 bits, as a result, there are 74,649 nodes in total. A3, A7, A8 and A9 have the best capacity over all these previous methods that are based on DNA sequences, since they hide two bits per base, but the capacity of the proposed method outperformed the capacity of all these previous methods (A1-A11), which required the lowest number of bases (20,480 amino acids), due to hiding four bits for each amino acid. Therefore, the proposed method saves 50% bandwidth compared with the approaches in (A1-A11) for each communication that transmits 10 kilo Bytes between sender and receiver.

7.3 Comparisons the proposed method with other existing techniques in term of average BPN, payload, protein conserve functionality and cracking probability

As shown in Table 8, the best algorithms in term of BPN for data hiding that based on DNA sequences is in [21-23] for hiding 20K Bytes data, since it uses

Table 6. The results of using proposed method to hide 22,630 bytes (22,630 × 8 = 181040 bits) secret message within various tested protein sequences

Accession Number	No of amino acids	Capacity	Payload	BPN
TSR51413	54498	54498	0	3.32
UCB56787	45376	45376	0	3.99
CAH0927406	46359	46359	0	3.90
KAI4565070	51741	51741	0	3.50
KAI4552952	48746	48746	0	3.71
KAI4551092	47538	47538	0	3.80
KAA0715513	45354	45354	0	3.99
Average bpn				3.74

Table 7. The required generated length for the sequence after hiding a secret message that is applied by proposed method vs some other techniques based on DNA sequence

IDs	Approach	Type of Sequence	Secret data size		
			(10) KBs (81920) Bits	**(1164)** KBs (9535488) Bits	**(3000)** KBs (24576000) bits
A1	Shiu et al., 2010 [8],Substitution method	DNA	81,920	9,535,488	24,576,000
A2	Guo et al., 2012 [16]	DNA	156,848	17,542,634	52,545,224
A3	Taur et al., 2012 [17]	DNA	40,960	4,767,744	12,288,000
A4	Khalifa and Hamad, 2015 [18]	DNA	245,760	28,606,464	73,728,000
A5	Malathi et al., 2017 [19]	DNA	122,880	14,303,232	36,864,000
A6	Saha et al., 2019 [20]	DNA	74,649	16,124,313	36,279,705
A7	Mohammed et al., 2019 [21]	DNA	40,960	4,767,744	12,288,000
A8	Mohammed and Abdel-Razeq, 2020 [22]	DNA	40,960	4,767,744	12,288,000
A9	Nabi et al., 2021[23],Substitution method	DNA	40,960	4,767,744	12,288,000
A10	Sabry et al., 2019 [24]	DNA	87,149	10,144,137	26,144,681
A11	Hassan et al., 2022 [25]	DNA	245,760	28,606,464	73,728,000
PA	Proposed Method	Protein	20,480	2,383,872	6,144,000

Table 8. A comparison between the suggested algorithm with DNA-based algorithms

Research	Approach	Average Bpn	Payload	Functionality conserved?	Cracking Probability
Ref [8]	Insertion Method	0.58	$\frac{\|M\|}{2}$	No	$\frac{1}{1.63\times10^{8}}\times\frac{1}{n\text{-}1}\times\frac{1}{2^{m}\text{-}1}\times\frac{1}{2^{s\text{-}1}}\times\frac{1}{24}$
	Complementary Method	0.07	$\|M\|(k+3\frac{1}{2})$	No	$\frac{1}{1.63\times10^{8}}\times\frac{1}{24^{2}}$
	Substitution Method	0.82	0	No	$\frac{1}{1.63\times10^{8}}\times\frac{1}{6}$
Ref [16]	Complementary Method	0.587	0	No	$\frac{1}{1.63\times10^{8}}\times\frac{1}{6}\times\frac{1}{24}$
Ref [17]	TLSM Method	1.64	0	No	$\frac{1}{1.63\times10^{8}}\times\frac{1}{24^{4}}\times\frac{1}{r}$
Ref [18]	LSBase	0.333	0	Yes	Not defined
Ref [19]	Insertion Method	1.52	$\frac{\|M\|}{2}$	No	$\frac{1}{1.63\times10^{8}}\times\frac{1}{24}\times\frac{1}{n\text{-}1}\times\frac{1}{2^{m}\text{-}1}\times\frac{1}{2^{s\text{-}1}}\times\frac{1}{2^{8m}}$
Ref [20]	Balanced Tree	0.97	Not defined	No	Not defined
Ref [21]	self-adaptive DNABS	2	0	Yes	$\frac{1}{2}\times\frac{1}{1.63\times10^{8}}\times\frac{1}{24}\times\frac{1}{m\times n^{2}\times p\times R^{2}}$
Ref [22]	Substitution Method	2	0	No	$\frac{1}{1.63\times10^{6}}\times\frac{1}{y\text{-}1}\times\frac{1}{2^{m-1}}\times\frac{1}{24}\times\frac{1}{L(y\text{-}1)}$
Ref [23]	Substitution Method	2	0	No	$\frac{1}{1.63\times10^{8}}\times\frac{1}{24}$
Ref [24]	Substitution Method	0.94	$N-\frac{101}{128}\times\|M\|$	Yes	$\frac{1}{1.63\times10^{8}}\times\frac{1}{2^{N/3}}$
Ref [25]	Substitution Method	0.333	0	Yes	$\frac{1}{1.63\times10^{8}\times(n-6)\times4!\times4!\times4}$
Proposed Method	Permutation Method	4	0	Yes	$\frac{1}{np}\times\frac{1}{20!}\times\frac{1}{(20!)^{20}}\times\frac{1}{16}\times\frac{1}{(\|PSII\|)!}$

a DNA bases to hide on maximum only two bits, while the suggested method doubles the hiding capacity (average BPN) by employing a base to hide up to four bits. The payload of the proposed method is Zero, since no expansion in the faked protein sequence after the embedding process. Also, the proposed method preserves the functionality of the protein reference sequence, since the protein reference sequence (PSI) is used only to determine the amino acids that must be permuted with another amino acids in the faked generated protein sequence (PSII). Finally, according to Table 8, the proposed method has lowest cracking probability over all existing methods.

8. Conclusions

This paper presents a novel steganographic technique that uses a protein sequence as a cover. The proposed method consists of two modules: embedding and extraction. The embedding module starts by partitioning each byte of the secret message into two 4-bit parts, and then these parts are converted into decimal numbers. Finally, the decimal values of the secret message are randomly included in the cover protein sites using a pseudo random number generator (PRNG) for each cover protein base value instead of being sequentially embedded in the cover protein bases. It is considered that this random distribution increases system security. The proposed method works with a one-digit protein decimal coding rule (PDCR) and the byte of the message will be embedded in two bases of amino acids. In contrast, the extraction module reverses the embedding process stages to disclose the secret message.

The experimental results showed a unique high performance of the proposed method compared to all the existing techniques that depend on DNA sequences as a carrier medium to conceal the data in terms of capacity, BPN and cracking probability. The proposed technique succeeded in achieving higher capacity and hiding 4-bit per amino acid (BPN), which is the highest among all methods; therefore, the proposed method saves 50% bandwidth compared with the other existing technique. Furthermore, the proposed method has been shown to be resistant to brute force attacks, making it nearly impossible to extract the secret message because its cracking probability is close to zero. So, the proposed method provides a higher level of security. We believe that the proposed method and its unprecedented results open a wide door for new steganography research based on protein sequences.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

The concept, method, programs, formal analysis, validation, resources, data curation and writing-original draft preparation were all contributed by Radhwan Jawad Kadhim. Hussein K. Khafaji Supervised, revised, and edited the work.

References

[1] H. Antonio, P. W. C. Prasad, and A. Alsadoon, "Implementation of cryptography in steganography for enhanced security", *Multimed. Tools Appl.*, Vol. 78, No. 23, pp. 32721–32734, 2019, doi: 10.1007/s11042-019-7559-7.

[2] Y. Niu, K. Zhao, X. Zhang, and G. Cui, "Review on DNA cryptography", In: *Proc. of International Conference on Bio-Inspired Computing: Theories and Applications*, pp. 134–148, 2019.

[3] O. A. A. Harbi, W. E. Alahmadi, and A. O. Aljahdali, "Security analysis of DNA based steganography techniques", *SN Appl. Sci.*, Vol. 2, No. 2, 2020, doi: 10.1007/s42452-019-1930-1.

[4] A. Khalifa, "A secure steganographic channel using DNA sequence data and a bio-inspired XOR cipher", *Inf.*, Vol. 12, No. 6, 2021, doi: 10.3390/info12060253.

[5] M. M. Amrulloh and T. Ahmad, "Utilizing fuzzy logic in developing reversible data hiding method", *Int. J. Intell. Eng. Syst.*, Vol. 13, No. 5, pp. 327–336, 2020, doi: 10.22266/ijies2020.1031.30.

[6] I. J. Kadhim, P. Premaratne, P. J. Vial, and B. Halloran, "Comprehensive survey of image steganography: Techniques, Evaluations, and trends in future research", *Neurocomputing*, Vol. 335, pp. 299–326, 2019.

[7] P. Malathi and T. Gireeshkumar, "Relating the Embedding Efficiency of LSB Steganography Techniques in Spatial and Transform Domains", *Procedia Comput. Sci.*, Vol. 93, No. September, pp. 878–885, 2016, doi: 10.1016/j.procs.2016.07.270.

[8] H. J. Shiu, K. L. Ng, J. F. Fang, R. C. T. Lee, and C. H. Huang, "Data hiding methods based upon DNA sequences", *Inf. Sci. (Ny).*, Vol. 180, No. 11, pp. 2196–2208, 2010, doi: 10.1016/j.ins.2010.01.030.

[9] H. K. A. Khafaji and Z. M. Jameel, "A New Approach to DNA, RNA, and Protein Motifs

Templates Visualization and Analysis via Compilation Technique", *IOSR J. Comput. Eng.*, Vol. 19, No. 02, pp. 15–25, 2017, doi: 10.9790/0661-1902011525.

[10] R. Jiang, X. Zhang, and M. Q. Zhang, *Basics of Bioinformatics: Lecture Notes of the Graduate Summer School on Bioinformatics of China*, Vol. 9783642389, 2013, doi: 10.1007/978-3-642-38951-1.

[11] B. Alberts, *Molecular Biology of the Cell*, WW Norton & Company, 2017.

[12] E. Keedwell and A. Narayanan, "Intelligent Bioinformatics: The Application of Artificial Intelligence Techniques to Bioinformatics Problems", *Intell. Bioinforma. Appl. Artif. Intell. Tech. to Bioinforma. Probl.*, pp. 1–280, 2005, doi: 10.1002/0470015721.

[13] J. Pevsner, *Bioinformatics and Functional Genomics*, 2005, doi: 10.1002/047145916x.

[14] M. Morange, "Based on the article entitled 'Fifty Years of the Central Dogma", *Publ. J. Bio-Sciences*, Vol. 33, No. March, pp. 171–175, 2009.

[15] G. Edited, I. C. Gray, and M. R. Barnes, *FOR GENETICISTS Edited by*, Vol. 4. 2003.

[16] C. Guo, C. C. Chang, and Z. H. Wang, "A new data hiding scheme based on DNA sequence", *Int. J. Innov. Comput. Inf. Control*, Vol. 8, No. 1, A, pp. 139–149, 2012.

[17] J. S. Taur, H. Y. Lin, H. L. Lee, and C. W. Tao, "Data hiding in DNA sequences based on table lookup substitution", *Int. J. Innov. Comput. Inf. Control*, Vol. 8, No. 10, pp. 6585–6598, 2012.

[18] A. Khalifa and S. Hamad, "Hiding Secret Information in DNA Sequences Using Silent Mutations", *Br. J. Math. Comput. Sci.*, Vol. 11, No. 5, pp. 1–11, 2015, doi: 10.9734/bjmcs/2015/19561.

[19] P. Malathi, M. Manoaj, R. Manoj, V. Raghavan, and R. E. Vinodhini, "Highly Improved DNA Based Steganography", *Procedia Comput. Sci.*, Vol. 115, pp. 651–659, 2017, doi: 10.1016/j.procs.2017.09.151.

[20] P. Saha, L. Y. Pinky, M. A. Islam, and P. Akter, "Higher Payload Capacity in DNA Steganography using Balanced Tree Data Structure", *Int. J. Recent Technol. Eng.*, Vol. 8, No. 4, pp. 6551–6556, 2019, doi: 10.35940/ijrte.d8088.118419.

[21] M. H. Mohammed, B. H. Ali, and A. I. T. Mohamed, "Self-adaptive dna-based steganography using neural networks", *Inf. Sci. Lett.*, Vol. 8, No. 1, pp. 15–23, 2019, doi: 10.18576/isl/080102.

[22] M. H. Mohammed and A. A. Razeq, "Dna-based steganography using genetic algorithm", *Inf. Sci. Lett.*, Vol. 9, No. 3, pp. 205–210, 2020, doi: 10.18576/isl/090307.

[23] S. H. Nabi, P. Sarosh, S. A. Parah, and G. M. Bhat, "Information Embedding Using DNA Sequences for Covert Communication", pp. 111–129, 2021, doi: 10.1007/978-981-15-8711-5_6.

[24] M. Sabry, T. Nazmy, and M. E. Khalifa, "Steganography in DNA Sequence on the Level of Amino acids", In: *Proc. of 2019 IEEE 9th Int. Conf. Intell. Comput. Inf. Syst. ICICIS 2019*, pp. 317–324, 2019, doi: 10.1109/ICICIS46948.2019.9014843.

[25] S. Hassan, A. Muztaba, S. Hossain, and H. S. Narman, "A Hybrid Encryption Technique based on DNA Cryptography and Steganography", In: *Proc. of IEEE 13th Annu. Inf. Technol. Electron. Mob. Commun. Conf. (IEMCON), Vancouver, BC, Canada*, pp. 501–508, 2022.

[26] "National Center for Biotechnology Information", https://www.ncbi.nlm.nih.gov/search/all/?term=protein

[27] "Primitive root", https://en.wikipedia.org/wiki/Primitive_root_modulo_n.