Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2025 (43)** : 1–18.
ISSN-0037-8712
doi:10.5269/bspm.68821

DNA Encoding Scheme Based on Biological Operations

Sara Benatmane* and Djilali Behloul

ABSTRACT: In this paper, we propose a new approach in symmetric key cryptography which consists of two rounds of processing for delivering information security services (Authentication, Integrity, and Confidentiality). In the first round, a CSPRNG is generated using C# based on a secure seed then is coded using a biological DNA operation to be used as a random key for OTP. In the second round of processing, the XOR arithmetic operation is used as an encryption technique. Our proposed system is more secure compared to the conventional binary coding encryption technique because the symmetric key goes through the biological process which adds confusion and making it more difficult to decrypt the cipher data. Using a message digest produced by MD5, the suggested method also ensures integrity.

Key Words: DNA cryptography, Biological operations, Central dogma of molecular biology, One time pad, Modified D-H algorithm.

Contents

1. Introduction

When data is transmitted or stored, it is frequently protected using cryptography; a technique for encoding and decoding data that renders it inaccessible to unauthorized parts. Cryptography is linked to information security traits like authentication, secrecy, non-repudiation, and integrity, by developing and assessing procedures that are resistant to the influence of adversaries [17].

Information security professionals discovered that binary computers (digital computers), particularly in data storage and calculation operations, have a number of physical limitations. Therefore they concentrated on DNA computers (bio-molecular computers) and quantum computers [11]. DNA computing is expanding in the current time. It is offering a brand new data structure and evaluating methods for the parallel processing abilities of molecules. With the advancement of DNA computing, DNA cryptography got its beginnings.

One-time pad (OTP) is a key generation principle used with the stream ciphering technique, in the world of cryptography, OTP is referred to be a perfect secrecy algorithm. OTP is a type of encryption cipher that solely use the key or pad, its secret key is a lengthy string of bits that were selected at random. The confidentiality of the keys, which must be unpredictable, is essential to the OTP's security [16]. OTP have a major limitation in that a large key needs to be generated, transmitted, and saved by the parties communicating in total secrecy.

In practice, a pseudorandom generating key is used in place of this truly random key. It is generated from an initial seed during the encryption and decryption processes using a sequence generator, such as a shift register with nonlinear feedbacks. The only part of this system that is kept secret is the seed,

* Corresponding author

Submitted July 05, 2023. Published November 04, 2023
2010 *Mathematics Subject Classification:* 94A60, 68P25, 92D20.

a relatively small number that is selected at random to describe the beginning state of the sequence generator [15].

Data is hidden using DNA sequences in cryptography, which can be accomplished using a variety of biological technologies. In recent years, the use of data hiding techniques to transmit secret information has become more widespread. Methods for data hiding based on DNA sequences have drawn a lot of attention, they showed a capability to thwart malicious entry and guarantee a secure transfer. Therefore numerous DNA-based cryptography approaches were presented to address the problem of data hiding. In [1] a two stage encryption scheme based on DNA sequence has been shown. By generating a random key in the first stage, plaintext is encrypted. In the second stage, the plaintext is once more encrypted to create the ciphertext. Additionally, the inventors of this encryption algorithm used a shared key to both encrypt and decrypt the intended message, based on a symmetric key cryptography scheme. Two steps are maintained to encrypt the original key. Such a key is sent over a different secure channel from the one being used to transport the ciphertext. According to [11], a cipher algorithm utilizing mathematical operations and biological phenomena was devised. Based on the conversion of DNA to RNA, a symmetric key generation system is created using biological processes. The arithmetic procedure utilized as an encryption method is the XOR operator. The suggested algorithm is a digital computer simulation of a biological function that might not fully take advantage of biological composition in terms of storage and parallel processing.

A revolutionary cryptosystem for safe data transfer was proposed in [12], the plaintext is subject to different cryptographic operations, and it is then transformed into a DNA sequence using a 256-bit secret key. The DNA sequence is coded using a Mealy machine created randomly, to increase the security of the ciphertext. The suggested system provides security against a number of intrusions. The approach provided in [10] proposes a DNA cryptographic method based on a dynamic DNA sequence table together with OTP, which enhances the level of data security. In order to compromise the security of the ciphertext, the attacker must first conduct every conceivable combination of checks, which is thought to be virtually impossible. For this reason, the suggested technique for decrypting data uses a dynamic DNA sequence table, random DNA sequence characters, and an OTP with a randomly generated key.

In this research, we suggest simulating DNA biological processes on a digital computer for OTP encryption. A CSPRNG in $C\#$ is used to generate a random binary key based on a secret seed, the binary key is then coded using the biological processes of DNA known as (the central dogma of molecular biology, mutation process) to produce an artificial DNA, protein, and sequence caused by mutation, which is then used as a random (DNA, Protein, Mutated) key. The suggested method makes use of the biological effects on DNA in order to generate secure cryptographic keys for symmetric ciphering. The NSIT test suit is used to evaluate the randomization performance of the key generation.
The next sections are organized as follows: section 2 provides biological background information in order to better comprehend the biological ideas underlying our algorithm. Section 3 describes the proposed algorithm, the key generation method, integrity and confidentiality service. Section 4 contains the implementation and GUI, it explains the algorithm components and forms design and functions for each component. Section 5 involves discussions and analysis about the algorithm and the randomness test of the generated key and compare the encryption and decryption time for the three methods of coded key in our algorithm. Section 6 presents the main conclusions that summarize the entire work, in addition to the future work suggestions.

2. Preliminaries

DNA in cryptography:

The fundamental component of the human body and the genetic code of all living organisms is the biological molecule DNA, abbreviated from Deoxyribo Nucleic Acid. It is particular to each person, present in every cell of the body, and is crucial to the identity of any living organism [2]. The identification of DNA as the primary genetic molecule that contains all the inherited information in the chromosomes has immediately provided a hint about its structure.
James Watson created the first 3D image of DNA on an X-ray print in 1953, showing that DNA molecules are made up of two biopolymer strands that coil around one another inside of cells to form a double

stranded helix that resembles to a spiral ladder [8]. Each strand of ssDNA produced by the division of dsDNA is known as a polynucleotide as it is made up of numerous smaller units known as nucleotides [5]. There are three main parts that make up each nucleotide:

(a) 5-carbon pentose sugar,

(b) Phosphate group,

(c) Nitrogenous base.

The nitrogenous bases are Guanine (G), Adenine (A), Thymine (T), Cytosine (C) and Uracil (U) [3]. The nitrogenous bases in DNA can be either purines or pyrimidines in kind. Cytosine (C) and Thymine (T) are classified as pyrimidines and Adenine (A) and Guanine (G) are termed purines. Pyrimidines only have one ring, whereas purines have two rings.

Biological operations

Hybridization (Anneal): In a biological process called hybridization or anneal, single stranded DNA chains or sequences are joined with other single stranded DNA to create complementary base pairs, which are the building blocks of double stranded DNA [6].

Transcription: During transcription, an enzyme separates the two strands of double-stranded DNA. A single-stranded messenger RNA (mRNA) is then synthesized by mapping the DNA sequence onto one of the split DNA strands. The corresponding DNA sequence for RNA bases (U, A, G, C) is (T, A, G, C), where uracil (U) in RNA replaces thymine (T) in DNA. Exons, which are the segments encoding information for protein synthesis and assembly, are retained and processed, while non-coding portions called introns are removed through splicing during transcription [14].

Translation: The transformation of DNA molecules into protein sequences is the central dogma of molecular biology. The three letter codes that make up genetic code are known as codons, and both DNA and RNA posses them. Transcription and translation are the two steps involved in converting DNA into proteins. The first stage of transcription converts DNA molecules to mRNA. The translation is the second stage, which converts an mRNA sequence into a protein [4]. The process of DNA transcription and translation, which results in the conversion of DNA into amino acids to create protein cells. DNA is only ever duplicated in one strand. A single gene may undergo thousands of transcriptions. The DNA strands rejoin after transcription to create amino acids, which are then converted into protein.
DNA contains all of the genetic instructions that are required for an organism to take on form and function. DNA nucleotide order encodes genetic information. Three nucleotide bases make up a codon, and different codon combinations can result in different amino acids, which can then result in distinct proteins [18]. So, in order to make a protein, the codon order is crucial.

Mutation: A mutation is a change in the genetic code [18]. It is brought on by either internal sources, such as possible mistakes in genetic material replications, or external factors, such as chemicals and radiations. The form of mutation known as a gene mutation takes place within a DNA gene. Gene mutations appear in three different forms:

A) Substitution: Is a modify in a single base pair by replacing one base for another base, meaning instead of match $A-T$, the wrong match is $A-G$. Three different kinds of substitution mutations exist [6].

   (i) Silent mutation, in which the amino acid is unaffected by the translation process.

   (ii) Nonsense mutation generates a stop codon amino acid, which results in an early termination of translation.

   (iii) A missense mutation that changes the amino acid that a codon specifies.

B) Insertion: occuring when one or more bases are introduced to the DNA sequence.

C) Deletion: when one or more nucleotides are deleted from the DNA sequence.

3. Main Results

The proposed algorithm:

This section describes the proposed DNA cryptography algorithm, which provides three key information security services: authenticity, integrity, and confidentiality. An upgraded Diffie-Hellman protocol enhanced with Zero-Knowledge Proof (ZKP) is used for authentication, enabling the secure exchange of the seed. The algorithm incorporates three operations for coded key generation: DNA, the central dogma of molecular biology, and mutation. Additionally, the integrity service utilizes the generated key along with the message to create a digital signature, while the confidentiality service employs the XOR function for encryption and decryption.

Authentication service:

With the help of a modified version of Diffie-Hellman that uses the zero knowledge proof protocol, the authentication service tries to establish secure communication between sender and receiver. This key will be used as a seed for key generation. To start the authentication process, two numbers p and g are chosen and announced as public numbers where p (modulo) is a large prime number and g is a primitive root of order $(p-1)$ in the group $\langle Z_p^*, x\rangle$ [9]. Both the sender and the receiver have access to these values. D-H authentication is described in algorithm 1 below.

Algorithm 1 Modified Diffie-Hellman key exchange algorithm [9]

Step 1: Declare variables
p= prime number
g= primitive root of $\mathbb{Z}_p$.
Step 2: Sender chooses a random number x, such that $0 < x < p$
Sender calculates R_1
$R_1 = g^x mod\ p$, then sends R_1 to receiver.
Step 3: Receiver chooses y, such that $0 < y < p$
Receiver calculates R_2, K_1 , C_1
$R_2 = g^y\ mod\ p$
$K_1 = R_1^y\ mod\ p$
$C_1 = E(K_1, R_2)$, then sends C_1 and R_2 to sender.
Step 4: Sender calculates K_2, R_2'
$K_2 = R_2^x\ mod\ p$
$R_2' = D(K_2, C_1)$
Step 5: Sender checks
If $R_2 = R_2'$
Display receiver is honest
else
Display receiver is dishonest
Step 6: Sender calculates C_2
$C_2 = E(K_2, R_1|R_2)$
Step 7: Receiver calculates R_1'
$R_1' = D(K_1, C_2)$
Step 8: Receiver checks
If $R_1 = R_1'$
Display sender is accepted
else
Display sender is rejected

The key generation step:

The sender generates a random binary key using the class cryptographically secure PRNG in $C\#$ depending on the seed obtained from the authentication process. Then, the sender converts the key into DNA sequence as described in algorithm 2, or converts it into a protein chain as described in algorithm 3, or applies to the key the stages of the mutation process as described in algorithm 4 to acquire the encoded form of the key generation.

Algorithm 2 DNA key generation algorithm

Step 1: Declare variables
M= Plaintext
Seed= Key from authentication process
Step 2: Calculate length(M)
Step 3: For $i = 0$ to length(M)
OTP-R(seed)= Generate random number from $[0, 1]$
End For
Step 4: Calculate length(OTP-R)
Step 5: For $n = 0$ to length(OTP-R)
Convert from binary to DNA-OTP
$\{A, C, G, T\} \leftarrow \{00, 01, 10, 11\}$
End For

Algorithm 3 Protein key generation algorithm

Step 1: Declare variables
M= Plaintext
Seed= Key from authentication process
Step 2: Calculate length(M)
Step 3: For $i = 0$ to length(M)
OTP-R(seed) = Generate random number from [0,1]
End For
Step 4: Calculate length (OTP-R)
Step 5: For $n = 0$ to length (OTP-R)
Convert from binary to DNA-OTP
$\{A, C, G, T\} \leftarrow \{00, 01, 10, 11\}$
End For
Step 6: Calculate length(DNA-OTP)
Step 7: Perform hybridization to create double strand DNA
For $k = 0$ to length(DNA-OTP)
Connect complementary base pairs
dsDNA$\leftarrow \{A, C, G, T\}, \{T, G, C, A\}$
End for
Step 8: Calculate length(dsDNA)
Step 9: Perform transcription to create mRNA
For f = 0 to length(dsDNA)
mRNA = $U \leftarrow T$
End for
Step 10: Calculate length(mRNA)
Step 11: Perform translation to create protein key
For $l = 0$ to length(mRNA)
Protein key = Amino-acid $\leftarrow$ mRNA
End for
Display protein key

Algorithm 4 Mutation key generation algorithm

Step 1: Declare variables
M= Plaintext
Seed= Key from authentication process
Step 2: Calculate length(M)
Step 3: For $i = 0$ to length(M)
OTP-R(seed)= Generate random number from $[0, 1]$
End for
Step 4: Calculate length(OTP-R)
Step 5: Convert OTP-R to DNA-OTP
For $n = 0$ to length(OTP-R)
Convert from binary to DNA-OTP
$\{A, C, G, T\} \leftarrow \{00, 01, 10, 11\}$
End for
Step 6: Calculate length(DNA-OTP)
Step 7: Perform hybridization to create double strand DNA
For $k = 0$ to length(DNA-OTP)
Connect complementary base pairs
dsDNA$\leftarrow \{A, C, G, T\}, \{T, G, C, A\}$
End for
Step 8: Calculate length (dsDNA)
Step 9: Perform transcription to create mRNA
For $f = 0$ to length(dsDNA)
mRNA = $U \leftarrow T$
End for
Step 10: Calculate length(mRNA)
Step 11: Perform mutation to create mutated mRNA
For $l = 0$ to length(mRNA)
Mutated mRNA$\leftarrow$ mRNA
End for
Step 12: Calculate length(Mutated mRNA)
Step 13: Perform translation to create mutated protein key
For $m = 0$ to length(Mutated mRNA)
Mutated protein key = Amino-acid $\leftarrow$ Mutated mRNA
End for
Display mutated protein key

Integrity service:

The integrity service is employed to make sure that the message transmitted from the sender to the receiver is unchanged. A 128 bits long digital signature that is applied to the actual message before it is encrypted in the following phase is the outcome of the integrity process. The integrity is described in the following algorithm 5.

Algorithm 5 Integrity service

Step 1: Declare variables
M= Plaintext
K= Generated key from key generation
i= number of blocks
Step 2: Split the message into blocks of length 128 bits (16 characters)
$M = M_1, M_2, M_3, M_4, \ldots\ldots M_i$
Step 3: Convert the blocks of message to binary
For $k = 0$ to i
$ASCII.M_i = ASCII.Getbyte(M_i)$
$Binary.M_i = ASCII.M_i$
End for
Step 4: Split the generated key into parts of length 128 bits
Step 5: Convert the key to binary
For $l = 0$ to 128
$ASCII.K = ASCII.Getbyte(K)$
$Binary.K = ASCII.K$
End for
Step 6: XOR operation between the blocks of message and the key
For $n = 0$ to i
D = Digital signature
$D_{i+1} = M_i \bigoplus K$
End for
Display D
Step 7: Calculate DL
DL= Length(D)
Step 8: Convert the digital signature to DNA sequence
For $m = 0$ to DL
DDNA= $\{A \leftarrow 00, C \leftarrow 01, G \leftarrow 00, T \leftarrow 11\}$
End for
Step 9: Add the resulted digital signature to the real message
Final message= M+DDNA

Confidentiality service:

The processes of encryption and decryption guarantee confidentiality. The XOR operation is the method used for encryption and decryption. The final message from integrity is XORed with the produced key from key generation to produce encrypted text. The encryption procedure is described in algorithm 6 below.

Algorithm 6 Encryption algorithm

Step 1: Declare variables
M= Final message from integrity
K= The generated key from key generation
Step 2: Calculate length(M), length(K)
Step 3: Convert the message to binary
For $k = 0$ to length(M)
ASCII-M = ASCII.Getbyte(M)
BiM= Convert.ToBinary(ASCII-M)
End for
Step 4: Calculate length(BiM)
Step 5: Convert the key to binary
For $n = 0$ to length(K)
ASCII-K = ASCII.Getbyte(K)
BiK = Convert.ToBinary(ASCII-K)
End for
Step 6: Calculate length(BiK)
Step 7: Calculate Pad-L
Pad-L= Max(Len1,Len2)
Step 8: Padding left side with zeroes to be equal
Pad-BiM= PadLeft(Pad-L,0)
Pad-BiK= PadLeft(Pad-L,0)
Step 9: XOR operation between the blocks of message and the key
For $i = 0$ to Pad-L
C= Pad-BiM $\bigoplus$ Pad-BiK
End for
Display C
Step 10: Calculate length(C)
Step11: Convert the cipher text to DNA sequence
For $m = 0$ to length(C)
C-DNA= $\{A \leftarrow 00, C \leftarrow 01, G \leftarrow 00, T \leftarrow 11\}$
End for
Display C-DNA

4. Experimental Results

The algorithm was implemented using Microsoft Visual Studio 2010 (Visual C#). The DNA cryptography algorithm includes three methods for transformation key generation: the DNA key, the protein key, and the mutation key. Each method provides specific security services, with corresponding implementation windows available on both the sender and receiver sides.

Implementation of DNA cryptographic algorithm:

The DNA cryptographic algorithm is designed with windows forms (GUI). There are four primary buttons on the system main form that deal with the system's parts. They are:

1 Home button.

2 DNA key algorithm button.

3 Protein key algorithm button.

4 Mutation key algorithm button.

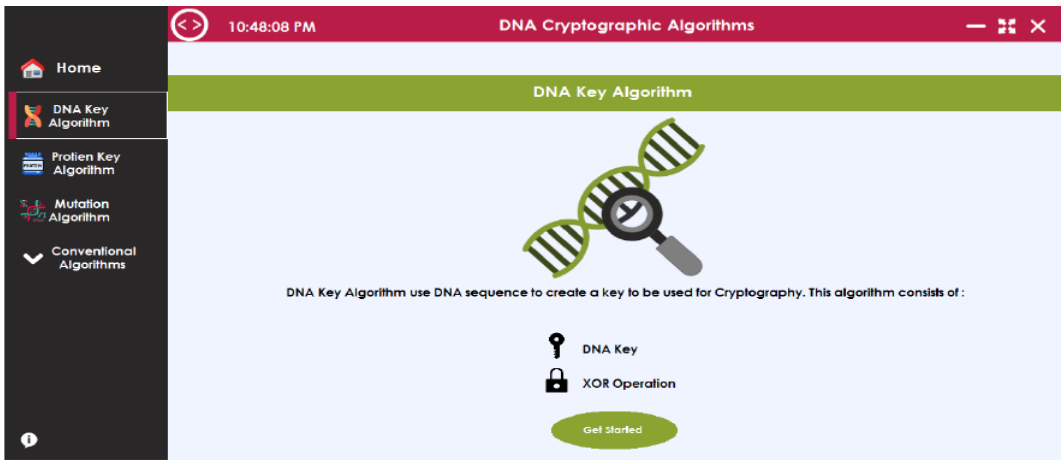

Figure 1: DNA key algorithm

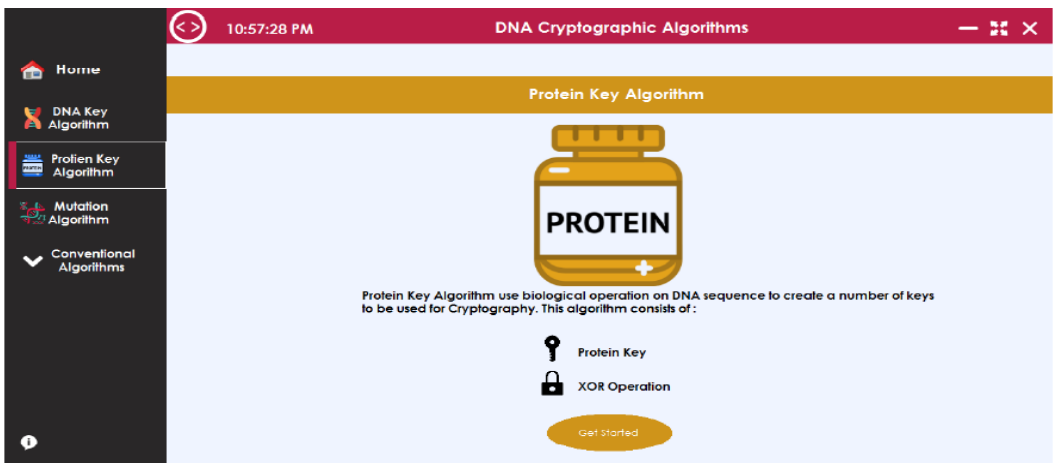

Figure 2: Protein key algorithm.

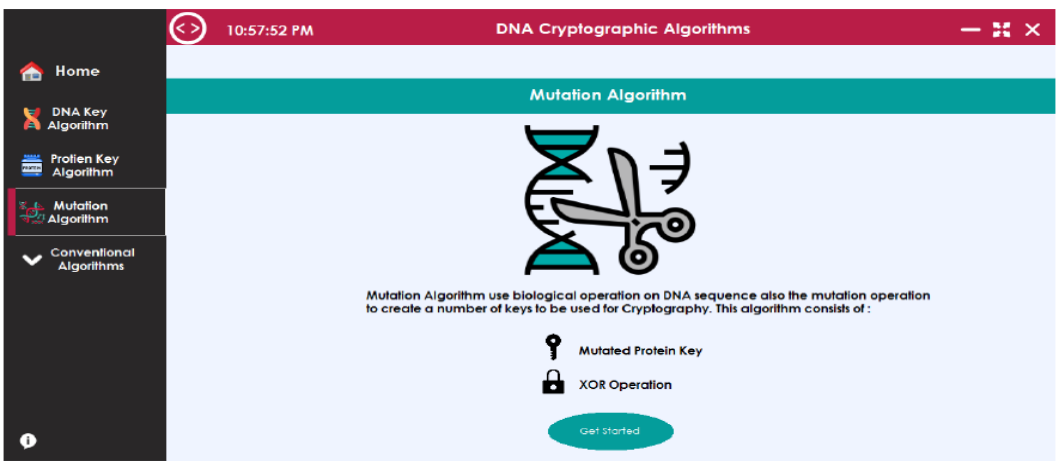

Figure 3: Mutation key algorithm.

Proposed system design:

Authentication service: The key exchange method known as modified Diffie-Hellman which is modified with zero knowledge proof (ZKP) was utilized by the authentication service to supply a key used as a seed in the binary random key generation.

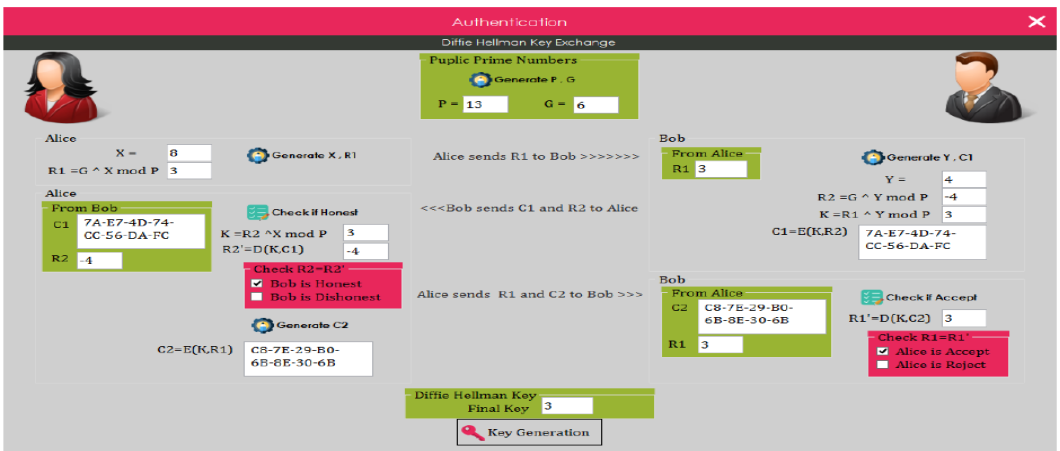

Figure 4: Authentication service.

Key generation: The sender selects one of three ways to coded the key generation for each encryption process. As part of the authentication procedure, the Diffie Hellman key is generated and is used as the seed to generate binary key, then coded to (DNA or protein key or mutated key). As depicted in Figures 5, 6 and 7.

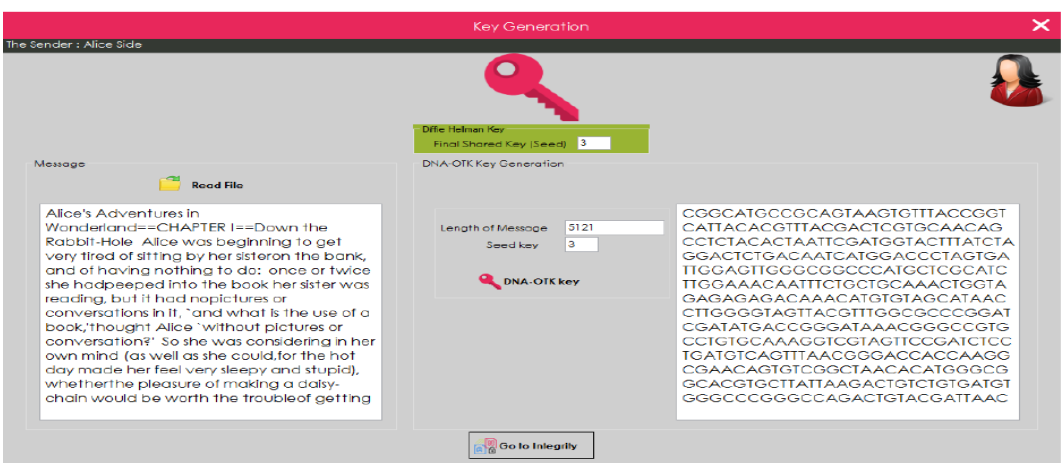

Figure 5: Key generation of DNA key algorithm.

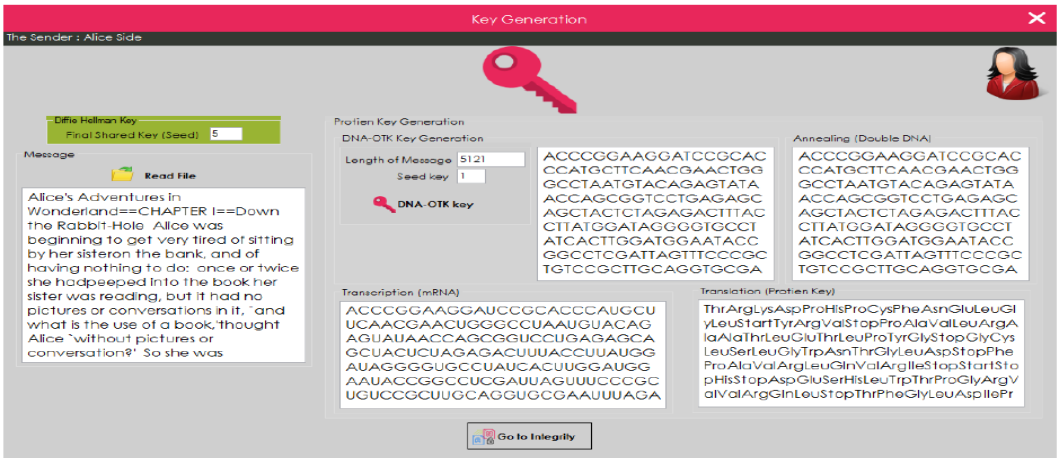

Figure 6: Key generation of protein key algorithm.

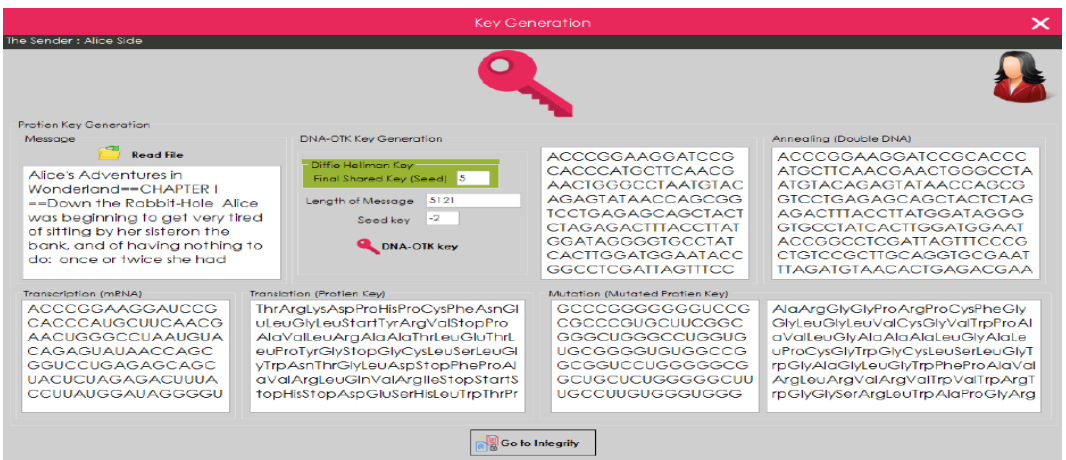

Figure 7: Key generation of mutation algorithm.

Integrity service: As illustrated in Figure 8, the integrity procedure divide the message into blocks of 128 bits long as well as took the key generated and split only 128 bit long from the key to generate a digital signature of 128 bits long. This digital signature is then added to the real message to be encrypted in the next step.

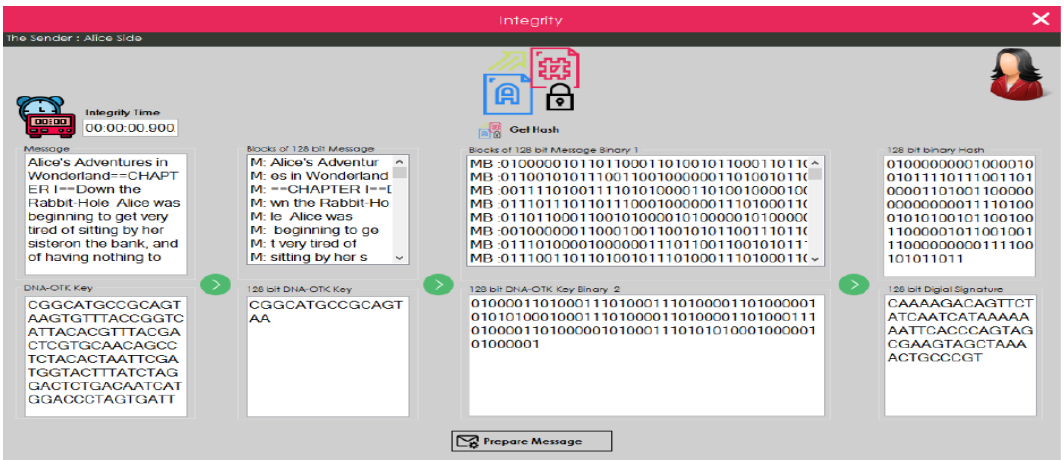

Figure 8: Integrity service of DNA key algorithm.

Confidentiality service: As shown in Figure 9 below, the plaintext is concatenated with the generated 128 bits hash message from integrity, and the result is then XORed with the generated key to create the encrypted message.

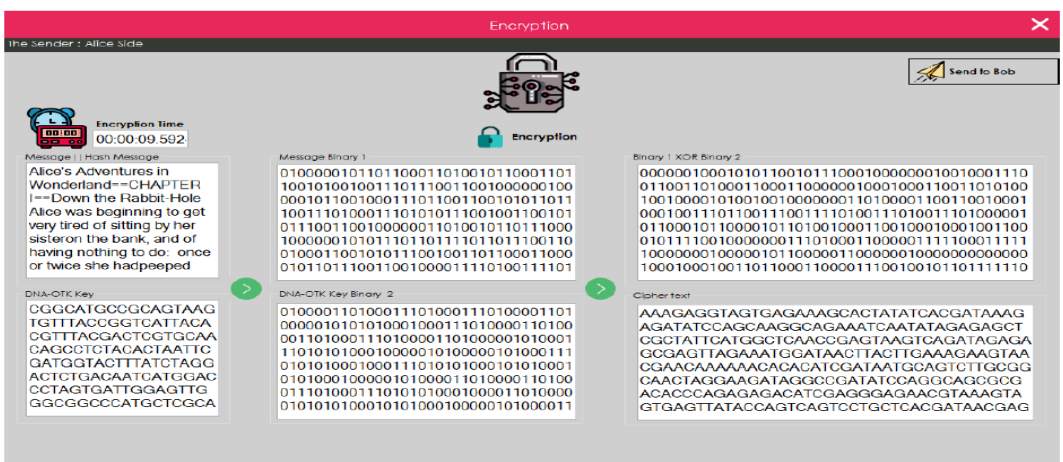

Figure 9: Encryption process of DNA key algorithm.

The identical seed from the authentication procedure is used to generate key on the receiver side, and as shown in figure 10, the cipher text then enters the decryption process to acquire the message from the sender. As seen in Figure 11, the recipient side checks the message's integrity.

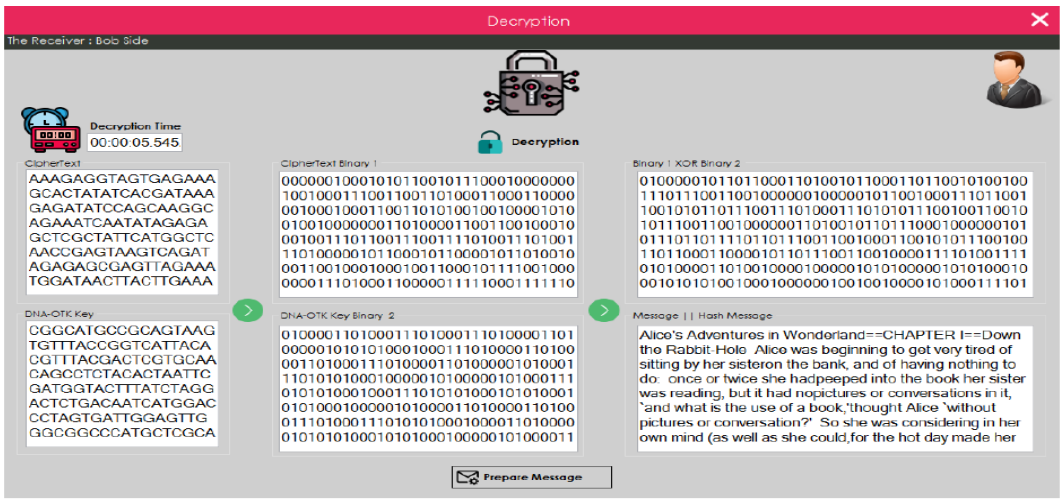

Figure 10: Decryption process of DNA key algorithm.

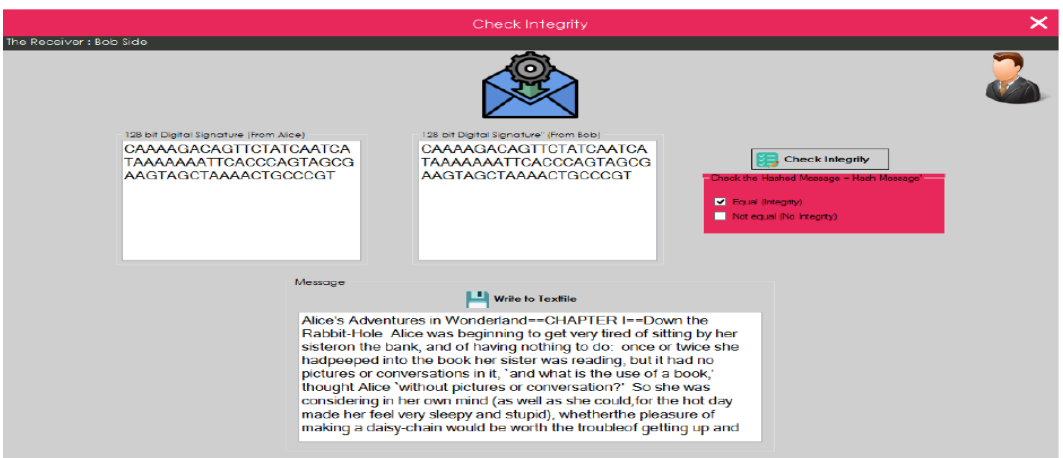

Figure 11: Checking the integrity of DNA key algorithm in receiver side.

5. System Analysis and Results

Randomness test: A randomness test involves assessing the randomness of the DNA-OTP key's sequences. Tests were advised by the NIST (National Institute of Standards and Technology) [13,7]. These tests concentrate on different types of randomness that would not occur in created strings or series. Every test has to fall within the 1% palette ticks range according to the guideline. The string is regarded as non random if the rule is less than 0.01. If not, it will be taken as random. Three proposed techniques, as shown in tables: Table 1, Table 2, and Table 3, employ a text file of 50 KB in order to evaluate the randomness test.

Table 1: Randomness test for key of DNA key algorithm

No	Test type	P-value	State
1	Frequency	0.534146	Pass
2	Block frequency	0.066882	Pass
3	Cumulative sums	0.739918	Pass
4	Runs	0.213309	Pass
5	Longest run	0.350485	Pass
6	FFT	0.122325	Pass
7	Non overlapping	0.534146	Pass
8	Overlapping template	0.350485	Pass
9	Maurer's test	0.350485	Pass
10	Serial test	0.534146	Pass

Table 2: Randomness test for key of protein key algorithm

No	Test type	P-value	State
1	Frequency	0.534146	Pass
2	Block frequency	0.122325	Pass
3	Cumulative sums	0.350485	Pass
4	Runs	0.739918	Pass
5	Longest run	0.911413	Pass
6	FFT	0.035174	Pass
7	Non overlapping	0.122325	Pass
8	Overlapping template	0.350485	Pass
9	Maurer's test	0.350485	Pass
10	Serial test	0.213309	Pass

Table 3: Randomness test for key of mutation key algorithm

No	Test type	P-value	State
1	Frequency	0.534146	Pass
2	Block frequency	0.122325	Pass
3	Cumulative sums	0.350485	Pass
4	Runs	0.739918	Pass
5	Longest run	0.911413	Pass
6	FFT	0.035174	Pass
7	Non overlapping	0.122325	Pass
8	Overlapping template	0.350485	Pass
9	Maurer's test	0.350485	Pass
10	Serial test	0.213309	Pass

According to the test when the values are more than 0.01, as demonstrated in the preceding tables, the key generation each time is random.

Time analysis:

The execution times for encryption and decryption as well as the time needed for the integrity process are measured in order to assess the system's running speed. The resultant information's throughput of encryption and decryption is computed as:

$$Encryption\ Throughput(ET) = Text\ file\ size/Encryption\ time$$

As indicated in the tables below, the analysis uses five text files of various sizes to compare the different methods.

Table 4: Comparison the encryption time between the different methods

Text file	Size in byte	Encryption time in millisecond (ms)		
		DNA key	Protein key	Mutation key
10KB	10,327	63,369.2178	143,387.3176	137,955.5725
20KB	20,509	129,311.3741	358,633.1413	343,461.4524
30KB	30,777	287,618.6943	823,824.5729	811,220.9542
40KB	41,018	486,055.5484	1,526,361.2073	1,522,006.5546
50KB	51,286	1,145,907.1765	3,716,519.2089	3,496,306.2012
Average		422,452.40222	1,313,745.0896	1,262,190.14698

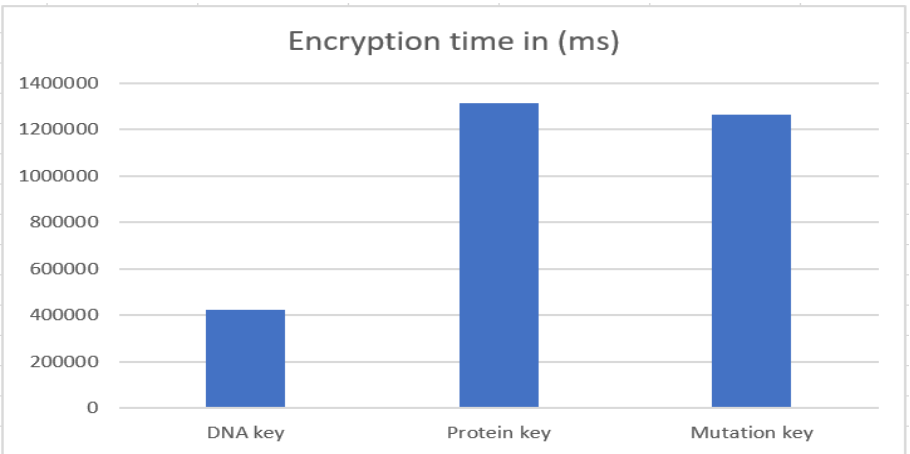

Figure 12: Average of the encryption time of the different methods.

Applying our proposed algorithm with the protein coded key, encryption time takes more than other coded schemes, as indicated in Table 4 and the flow chart in Figure 12. Therefore the execution times of the three suggested DNA coded key generations differ. This is due to the biological process that alters the DNA sequence which renders the created key difficult and time-consuming to encrypt. In reality, DNA biological processes are carried out in parallel in laboratories. Additionally, a molecular computer with low power requirements was needed since chemical interactions between the nucleotides are used to build the lengthy DNA strings, and these chemical bonds require a power source to form.

Table 5: Comparison of the encryption throughput between the different methods

Text file	Size in bytes	Encryption throughput in (b/ms)		
		DNA key	Protein key	Mutation key
10KB	10,327	0.162965	0.072021	0.074857
20KB	20,509	0.158601	0.057186	0.059712
30KB	30,777	0.107006	0.037358	0.037939
40KB	41,018	0.084389	0.026873	0.026949
50KB	51,286	0.044755	0.013799	0.014668
Average		0.111543	0.041447	0.042825

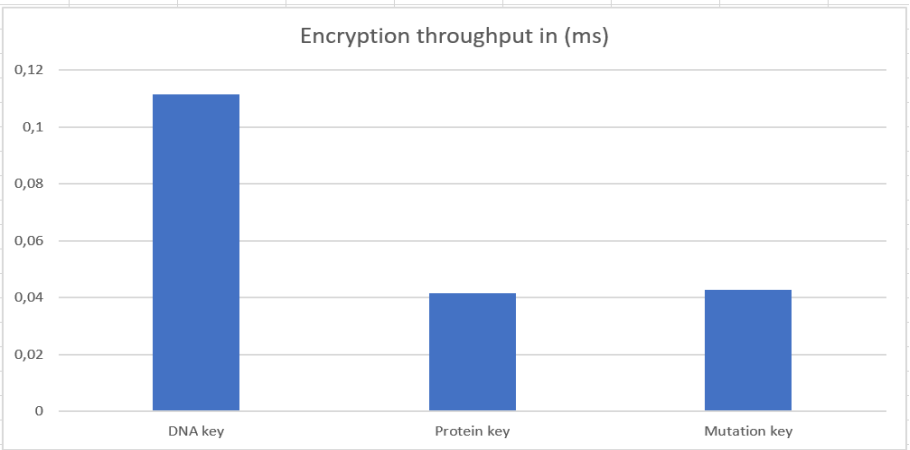

Figure 13: Average of the encryption throughput of the different methods.

Notice that the three proposed coded key generation algorithms in our approach has a distinct speeds, as indicated in Table 5 and the flow diagram in figure 13. This is due to the biological operation on the DNA sequence, which made the generated key in the three algorithms complex and time consuming in the encryption process. DNA biological operations are performed in parallel and the computation speed can reach 1 billion operations per second.

Table 6: Comparison of the decryption time between the different methods

Text file	Size in byte	Decryption time in millisecod (ms)		
		DNA key	Protein key	Mutation key
10KB	10,327	42,821.1993	136,771.4463	127,430.3904
20KB	20,509	85,576.5756	316,344.4398	309,979.0025
30KB	30,777	191,483.741	807,264.7008	653,469.105
40KB	41,018	311,399.7553	1,411,457.784	1,328,475.501
50KB	51,286	871,555.6491	3,641,400.8356	3,212,358.996
Average		300,567.38406	1,262,647.8413	1,126,342.59898

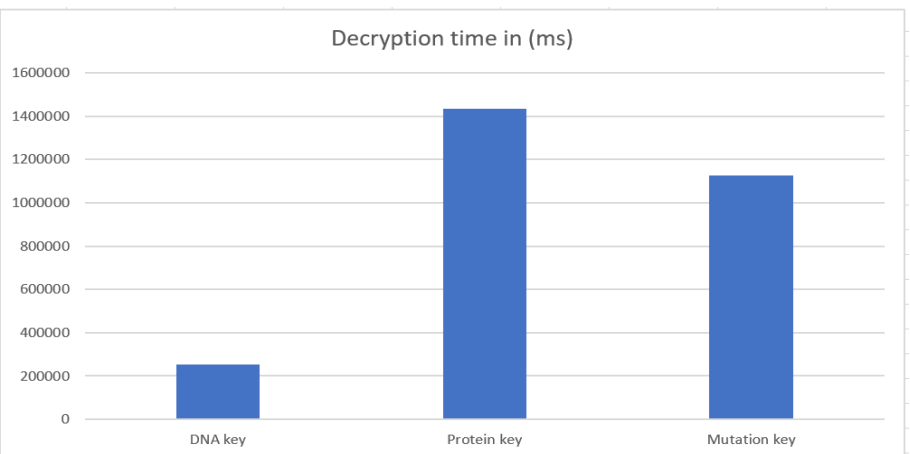

Figure 14: Average of the decryption time between the different methods.

Table 7: Comparison of the decryption throughput between the different methods

Text file	Size in bytes	Decryption throughput in (b/ms)		
		DNA key	Protein key	Mutation key
10KB	10,327	0.241165	0.075505	0.081040
20KB	20,509	0.239656	0.064831	0.066162
30KB	30,777	0.160729	0.038125	0.047097
40KB	41,018	0.131721	0.029060	0.030875
50KB	51,286	0.058844	0.014084	0.015965
Average		0.166423	0.044321	0.0482278

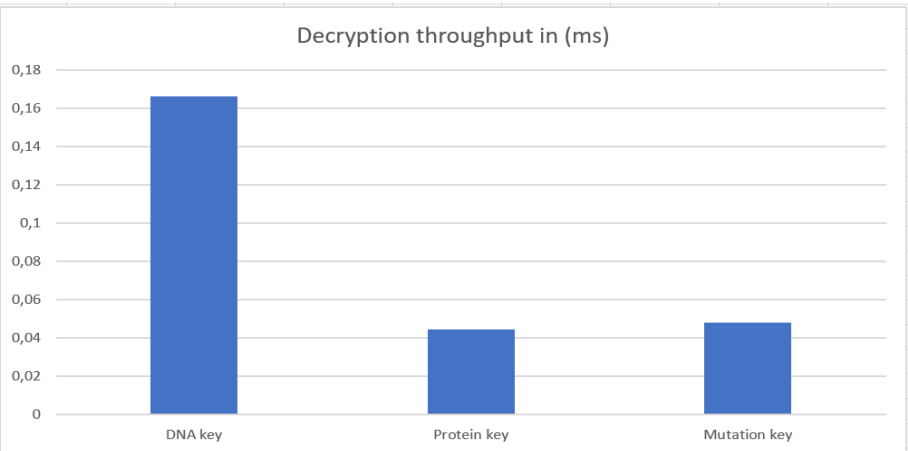

Figure 15: Average of the decryption throughput between the different methods.

6. Conclusion and Future Work

In order to provide information security services, this study suggested a DNA cryptography algorithm comprising three biological operations to transform the binary key generation. This kind of transformation is crucial because it presents data in a biological format that allows it to pass through biological processes and tests. Particularly those involving DNA and amino acids. In addition, it provides a means of observing data as it flows through biological processes and represents it in a binary format that is widely used in computer programs. Even if the proposed method ensure the security of the communication. The experimental results have shown that this method is time consuming for that we suggest the following conclusions:

- The key generation of the DNA key algorithm require a random seed to produce a random number generator to be converted to DNA sequence, this could be changed by using real DNA sequence.
- To create a protein key, the protein key algorithm uses several biological processes, including hybridization, transcription, and translation. By performing biological procedures in laboratories, the time needed to complete them can be decreased.
- The mutation algorithm can results an errors during the creation of mutated protein, so the researcher must be careful when performing this operation.
- Because biological operations take place in laboratories while the proposed algorithm are simulation processes using a digital environment, they cannot achieve the necessary properties in the biological environment such as storage and speed due to the parallel processing nature of biological operation of DNA, which take place in laboratories. This causes the encryption and decryption processes to

take different amounts of time with the proposed algorithm.
In order to improve the proposed system, some points can be suggested as a future work:

- Studying the possibility of using the real DNA sequence instead of pseudo DNA sequence that is generated using random number generators.
- More biological operations can be used in the key generation for improving the proposed algorithm such as executing other types of mutation on DNA sequence to generate different types of mutated proteins.
- DNA cryptographic algorithm can be combined with traditional cryptography such as AES and DES to provide hybrid security and enhance the performance of traditional cryptography.

References

1. A. Aich, A. Sen, S.R. Dash. A symmetric key cryptosystem using DNA sequence with OTP key. Information Systems Design and Intelligent Applications: Proceedings of Second International Conference INDIA, Volume 2. Springer India, (2015). 207 − 215.

2. S. AL-Wattar and al. Review of DNA and Pseudo DNA Cryptography, International Journal of Computer Science and Engineering, Vol 4, (2015), 65-76.

3. J. Chen, A DNA-based, biomolecular cryptography design. In 2003 IEEE International Symposium on Circuits and Systems (ISCAS) (Vol. 3, pp. III-III). IEEE, (2003, May).

4. S. Dhawan, A. Saini. Secure data transmission techniques based on DNA cryptography. Int. J. Emerg. Technol. Comput. Appl. Sci, vol. 2, no 1, p. 95-100 (2012).

5. A. Gehani, T. LaBean, J. Reif. DNA-based cryptography. Aspects of molecular computing: essays dedicated to tom head, on the occasion of his 70th birthday, (2004), 167 − 188.

6. O. Goldreich. Modern cryptography, probabilistic proofs and pseudorandomness. Springer science and business Media, Vol. 17, (1998).

7. R. Hegadi, A.P. Patil. . A Statistical Analysis on In-Built Pseudo Random Number Generators Using NIST Test Suite. In : 2020 5th International Conference on Computing, Communication and Security (ICCCS). IEEE, 2020. p. 1-6.

8. H.Z. Hsu, R.C.T. Lee. DNA based encryption methods. The 23rd workshop on combinatorial mathematics and computation theory, (2006, April), 545.

9. M. K. Ibrahem.(2012, April). Modification of Diffie-Hellman key exchange algorithm for Zero knowledge proof. In 2012 international conference on future communication networks (pp. 147-152). IEEE.

10. S. Kalyani, N. Gulati, Pseudo DNA cryptography technique using OTP key for secure data transfer. International journal of engineering science and computing, (2016) vol. 6, no 5, p. 5657-5663.

11. S.S. Nafea, M.K. Ibrahem, Cryptographic algorithm based on DNA and RNA properties. International journal of advanced research in computer engineering and technology, vol. 7, no 11, p. 804-811.(2018).

12. P. Pavithran, S. Mathew, et al. A novel cryptosystem based on DNA cryptography and randomly generated mealy machine. Computers and Security, vol. 104, p. 102160, (2021).

13. A. Rukhin, J. Soto, et al. A statistical test suite for random and pseudorandom number generators for cryptographic applications. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, vol 22, (2001).

14. M. Sabry, M. Hashem, T. Nazmy.Digital encoding to the form of amino acids for DNA cryptography and biological simulation. International Journal of Computer Applications, 2017, vol. 165, no 10.

15. A. Shamir. On the generation of cryptographically strong pseudorandom sequences. ACM Transactions on Computer Systems, vol 1, (1983), 38-44.

16. S. Tang, F. Liu. A one-time pad encryption algorithm based on one-way hash and conventional block cipher. 2nd International conference on consumer electronics, Communications and entworks, (2012, April). 72-74.

17. P.S. Varma, K.G. Raju. Cryptography based on DNA using random key generation scheme. International Journal of Science Engineering and Advance Technology, vol. 2, no 7, (2014). 168-175.

18. Q. Zhang, L. Guo, X. Wei. A novel image fusion encryption algorithm based on DNA sequence operation and hyper-chaotic system. optik international journal for light and electron optics, vol. 124, no 18, p. 3596-3600, (2013).

Sara Benatmane,
Faculty of Mathematics,
University of Sciences and Technology Houari Boumediene,
BP 32, El Alia, 16111, Bab Ezzouar, Algiers, Algeria.
LATN Laboratory, Faculty of Mathematics, USTHB, BP 32, El Alia, 16111, Bab Ezzouar, Algiers, Algeria.
E-mail address: benatmanesara34@gmail.com

and

Djilali Behloul,
Faculty of Computer Sciences,
University of Sciences and Technology Houari Boumediene,
BP 32, El Alia, 16111, Bab Ezzouar, Algiers, Algeria.
LATN Laboratory, Faculty of Mathematics, USTHB, BP 32, El Alia, 16111, Bab Ezzouar, Algiers, Algeria.
E-mail address: dbehloul@yahoo.fr