Journal of Theoretical and Applied Information Technology
31st May 2025. Vol.103. No.10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

NEXT-GEN SECURITY: LEVERAGING DNA CRYPTOGRAPHY FOR ROBUST ENCRYPTION

GURU PRAKASH B[1], SIVA T[2], SHUNMUGASUNDARAM S[3], MARIAPPAN E[4], ANNA LAKSHMI A[5], RAMNATH MUTHUSAMY[6]

[1]Department of Artificial intelligence and Machine Learning, Sethu Institute of Technology, Tamil Nadu, India.
[2]Department of Computer Science and Business Systems, Thiagarajar College of Engineering, Madurai, Tamil Nadu, India.
[3]Department of Information Technology, P.S.R. Engineering College, Sivakasi, Tamil Nadu, India.
[6]Department of Artificial Intelligence and Data Science, Ramco Institute of Technology, Rajapalayam, Tamil Nadu, India.
[5]Department of Information Technology, R.M.K. Engineering College, Thiruvallur, Tamil Nadu, India.
[6]Department of Artificial Intelligence and Data Science, Ramco Institute of Technology, Rajapalayam, Tamil Nadu, India.

*Corresponding author(s). E-mail(s): ramnath25@gmail.com;
Contributing authors: guru_netprakash@yahoo.co.in; tsca@tce.edu; shunmugammsec@gmail.com; mapcse.e81@gmail.com; annalaxmi.raj@gmail.com; ramnath25@gmail.com;
†These authors contributed equally to this work.

ABSTRACT

Cloud computing is the popular growing technology that provides services through the internet for data sharing and storage, access. Cryptography is the study of protecting the information by using algorithms, codes so that the intended users can view the data. Cryptography plays a vital role while transmitting data through networks and it's very important to ensure the confidentiality of the data. In this paper, to achieve and enhance the confidentiality of the data, DNA cryptography has been proposed. DNA cryptography is used to enhance the security of the data which is purely based on the nucleotide of DNA. The proposed modernized DNA cryptography algorithm is implemented using the .net framework and examples are also given with screenshots for the conclusion.

Keywords **-** *Cloud Computing, Secure Communication, DNA Cryptography, Amino Acid Tables, Dynamic Key Generation.*

1. INTRODUCTION

Cloud computing provides a different kind of services through the network that primarily involves databases, servers, data share, accessing of data, storage. It delivers service over the network to users so that they can access the data from anywhere at any cost. Secure communication is very important while delivering the service and we need to make sure that the data is accessed by the intended recipient. Now a day, the internet has become the media for all e-commerce platforms, e-healthcare, etc. Data communication must be made in a highly secure and efficient manner.

To cater to the security requirements, a lot of techniques and systems have been developed in the area of cryptography for making communication secure. We can enhance the strength of the cryptography methods by using DNA cryptography techniques. DNA cryptography is a modern art of science that uses nucleotides' of DNA and biological process for encoding. In this kind of computing, the encryption method is defined in terms of genetic coding instead of binary coding.

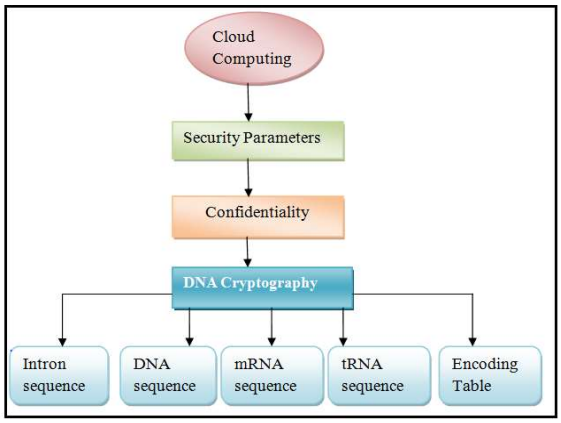

Fig.1.Overview of the cryptosystem

In DNA cryptography, the plaintext can be encrypted using DNA encoding tables based on DNA sequences, and it will undergo a lot of diffusion matrices to enhance data confidentiality in cloud computing. DNA cryptography provides new hope for the science of cryptography. Properties of DNA computing are also implemented while building the DNA algorithm, which might be useful to make our transmission secure in today's world.

In this paper, we introduce a new enhanced and secure DNA cryptosystem for the encryption and Decryption process. The various sections involved in this paper are as follows: Section 2 describes existing work in DNA cryptography, Section 3 describes the system architecture, Section 4 describes the proposed scheme, Section 5 describes the detailed example, Section 6 describes performance analysis of the proposed DNA cryptosystem and the conclusion is given in section 7.

2. RELATED WORK

Prema T. Akkasaligar (2020) proposed an encryption scheme based on a dual hyperchaos map and DNA cryptography that is for securing medical images and ensuring confidentiality. In the proposed scheme, permutation and diffusion are done and it's followed the encoding rules of the DNA biological process. To reduce the computational time, the selected pixels of the original medical image are shuffled while doing permutation. Dual hyperchaos map provides complex confusion property, which is to provide high-level security for digitalized medical images and entropy value is high. It shows the confusion property is handled efficiently.

Prasanna Balaji Narasingapuram (2020) introduced a symmetric key encryption technique for enhancing user-level security in cloud computing based on DNA cryptography. Initially, data is converted into ASCII code and it's converted into DNA based binary codes. The binary codes are represented in terms of nucleotides of DNA (A, T, C, G) using the DNA Binary code table and to increase the key strength, a random key generation table is brought into use which has the DNA code and key. To achieve user authentication User authentication, the key generation part is mainly focused, and user authentication is verified by using DNA cryptography-based key generation randomly.

Anupam Das (2019) explained the biological process of DNA and its nucleotides', codons. In this paper, the biological process is used to do the encryption which is purely based on DNA cryptography. This scheme has three modules for securing the data (i.e.) Key generation, encryption, and decryption. In the key generation module, the input password is given by the user, and it's converted into ASCII and then binary codes.
These binary codes are converted into nucleotides' and the transcription process is performed. Then, checking the stop codons and finding the longest key among them. Finally, in the encryption process, a circular left shift is performed on the key, and XOR operation has done on the message and key. But there is no dynamic DNA sequence table to increase the dynamicity of the key generation process.

A.Vikram (2019) brought up a new symmetric DNA cryptography algorithm to enhance the security of the data. In this 64-bit block cipher, the reverse binary coding process is followed to get fake DNA sequence, and final ciphertext is attained by applying Central Dogma of Molecular Biological (CDMB) process on the fake DNA sequence and it can be sent to the receiver in the form of protein sequence. In key generation process, 3 keys needs are derived (K1, K2, K3) from 96 base long DNA sequence that is the input given by sender. The first and last keys (K1 & K3) are used in encryption and the middle key (K2) is used for decryption.

Nagaraj (2018) suggested a DNA cryptosystem that is based on the DNA matrix. Initially, plaintext is converted into binary bits and it's formed as an 8*8 matrix. Zero will be appended during matrix formation in case of balancing the length of plaintext. As a next step, the resultant will undergo a spiral transformation, and then it will be moved back to binary bits, where the binary bits are converted into decimals by using the DNA coding table. In this paper, they haven't used any rotation or any kind of sting operations while doing encryption and decryption modules. The issue noticed here is that zeros are appended in order to balance the length, which will increase the length of the input and lead to more time for doing encryption, which might affect the algorithm performance.

Table.1. Analysis on various DNA Cryptosystem Models

S.No	Author, Year	Usage of Dynamic Encoding Table	Utilization of Full character set in encoding table	Unique sequence for each character	Strength of dynamic encryption
1	Prema T, Akkasaligar, 2020	No	No	No	Less secure
2	Prasanna Balaji, Narasingapuram, 2020	No	No	Yes- DNA binary code table is introduced	Less secure
3	A.Vikram, 2029	No	No	No	Computational time is high for huge amount of data
4	Hamza, Hammami, 2018	Yes	No	No	Time consuming process (matrix manipulation involved)
5	Our proposed cryptosystem	Yes	Yes (256 char)	Yes	Yes , more strength as it has dynamicity concept

3. SYSTEM ARCHITECTTURE

The data sender chooses the key and the plaintext for transmission, which will be subject to a secure DNA cryptosystem for processing. The key (intron sequence) can be selected using a random technique with a string length of ten. The data sender/owner generates two DNA sequences at random using the unique combination of DNA nucleotides Adenine (A), Thymine (T), Cytosine (C), and Guanine (G), which can be used to generate DNA encoding tables and then encrypted. The plaintext is encrypted using the DNA encoding table and undergoes a lot of string level rotation and bio-logical process encryption.

.

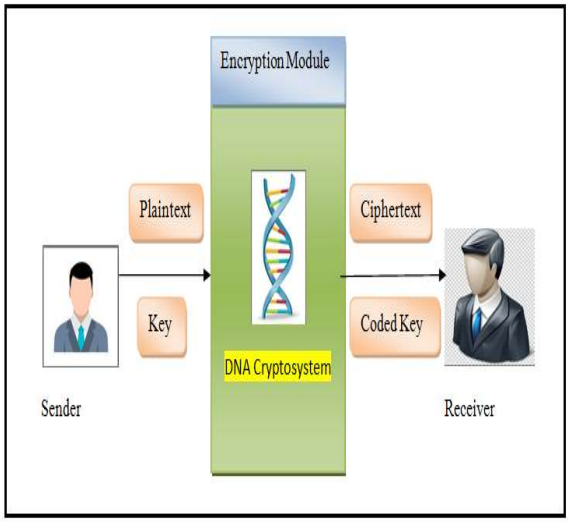

Fig.2. Block Diagram

The data receiver sends a request to the data owner/sender to get a key for decrypting the data. The key can be shared with the data user once he is authorized. Once the data user receives the encrypted key (coded key), the data user decrypts the key using the DNA algorithm and then retrieves the DNA sequence, which can be used for building the DNA encoding table. Finally, the data user can see the plaintext. Confidentiality of the data is enhanced using the random encoding table and encryption method.

The below open issue's were addressed in existing crypto system,

- ✓ More computational time is required for encrypting the data
- ✓ The Key size is very large which requires more space while data transmission and it needs more time for process execution
- ✓ The key block can be easily hacked by someone else in the middle environment as its not encrypted before sending to the data user

As per proposed DNA cryptosystem, during transmission, the cipher-text will be changed every time the same plaintext is used for processing, since a dynamic concept is involved while implementing the algorithm. Dynamic method can be achieved by generating the DNA encoding table for each transmission, which is based on the dynamic selection of the DNA sequence by the data sender. In the proposed architecture, a static table is not used, as we are generating the DNA encoding table dynamically for each transmission in the cloud. The intron sequence can be chosen by using below formulae which enhance the robustness of the key block.

Journal of Theoretical and Applied Information Technology
31st May 2025. Vol.103. No.10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Intron sequence selection (Key)

- ✓ Any four characters from alphabets,
- ✓ Any six digits from numerical

Initially, the sender begins the communication by choosing the plaintext that they want to send to the intended receiver. After that, he is ready to choose the key (intron sequence), which plays a vital role in the encryption process. The sender sends both the inputs to the DNA cryptography block where the main process starts for establishing secure communication.

In the DNA encoding block, the encryption process will be done by using the defined steps. The ciphertext and the key sequence will be stored in a central storage place. Then the sender informs the receiver to get the coded text. The receiver sends a request to the storage server. The server authenticates the receiver's identity. A signal will be given to the receiver once he is authorised by the server. If not, the receiver's request will be rejected by the server. After the successful authorization, the receiver will get the coded text and the key; he will do decryption and then form the plaintext accordingly.

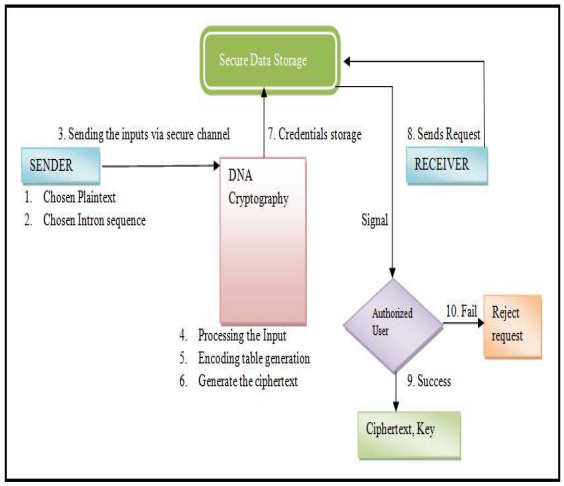

Fig.3. Work Flow Diagram

4. PROPOSED TECHNIQUE

The proposed system generates DNA-based ciphertext for ensuring the confidentiality of data while transmitting in the cloud. This modernistic scheme generates an encoding table dynamically using two DNA sequences which are given by the user, followed by the encryption process is done using an intron sequence (Randomly chosen string having a length of ten, 2 uppercase characters, 2 lowercase characters, 6 digits) and applied permutation and combinations rules. To increase the security of the information, random number and sequence generation logic is proposed. The main modules of the proposed scheme are given below.

A. Dynamic encoding table generation
B. Encryption Method
C. Decryption Method

A. Dynamic encoding table generation

Initially, 2 DNA sequences are randomly selected by the sender, and then the DNA sequence is converted into mRNA by replacing thymine with uracil. Then convert mRNA sequence into tRNA sequence. As a next step. The reverse process of the mRNA sequence is done. Now, 4*4 matrices are formed by assigning the mRNA sequences row-wise and column-wise randomly. Then it's extended into a 64*4 matrix table, and it frames Sequences of the DNA encoding table. To map the key codes to the mRNA sequences, the character set is used with random logic.

As follows, a total of 64 characters for each column, it's chosen as having Numbers (10), Upper case (26), Lower case (26), Special Characters (2), and $0, *1, ^2, @3 are added at the end of the code with each of the columns.

Finally, the mRNA sequence is mapped with key code in the encryption process. The flow diagram for the process of dynamic encoding table generation is shown in Fig.1.

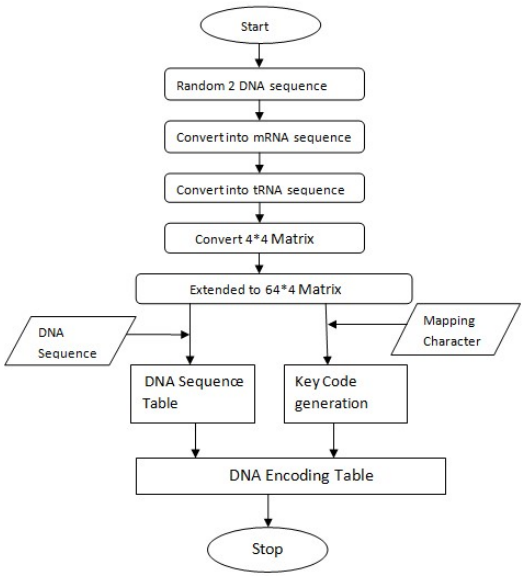

Fig.4. Flow diagram for dynamic DNA encoding table generation.

Journal of Theoretical and Applied Information Technology
31st May 2025. Vol.103. No.10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Encoding Table (DS_1, DS_2, Collate_Chars)
Input: DNA Sequence1 **DS_1**, DNA Sequence2 **DS_2**,
Output: Dynamic Encoding Table **ET**
Steps:
Convert **DS_1**, **DS_2** into mRNA sequence **DS_1 m**, **DS_2 m**
Convert **DS_1 m**, **DS_2 m** into tRNA sequence **DS_1 t, DS_2 t**
Generate **4*4** Matrix
Extended into **64*4** Matrix
Generate Key code (**64**)
Mapping of DNA sequence with key code
Final **Dynamic Encoding table**

B. Encryption Method

To begin with the encryption process, 2 random DNA sequence, Intron sequence is chosen and then original messages are given by the sender. Then, reverse all the inputs and convert them into ASCII codes. ASCII codes are finally converted into binary bits with the length of 8, append 0 as a prefix in cases where the string is not in the length of 8, the plaintext and intron sequences are not in the same length. now, split the binary code of the Intron sequence by its odd position and even position and same has been done on plaintext as well. Then the XOR operation has done on the odd position of the Intron sequence and the odd position of the plaintext.

Similarly, XOR operation has been done on the even position of the Intron sequence and the even position of the plaintext. The results have been concatenated and then 1-bit left circular shift operation has been done on the resultant. Then, the binary sequences are converted into DNA sequences using the DNA Nucleotide table, which is given in Table II. Then it's converted into an mRNA sequence by replacing thymine with uracil (T→U) and at last, mRNA sequence is converted into a tRNA sequence. The final tRNA sequence is mapped with the DNA encoding table, which is shown in Table III. The key codes are fetched from the dynamic DNA encoding table to make up the Ciphertext.

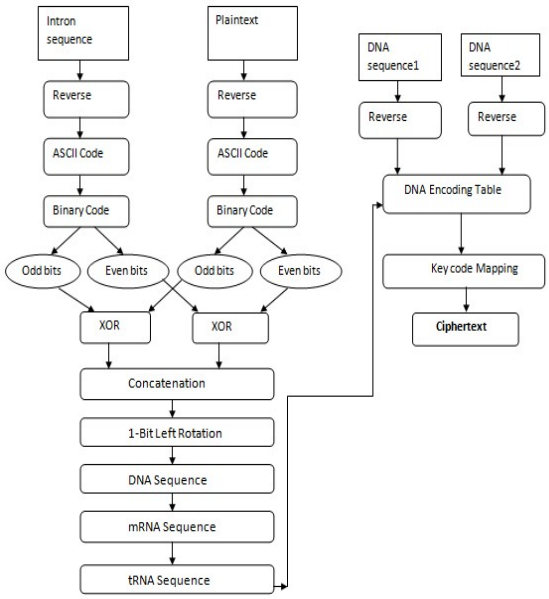

Fig.5. Flow diagram for encryption method

Table.2. Binary to DNA Nucleotide Mapping

Binary	DNA Sequence
00	A
01	C
10	G
11	T

DNA_Encryption_Method (pt, DS_1, DS_2, intr_seq)
Input: plaintext **pt,** DNA sequence1 **DS_1,** DNA sequence2 **DS_2,** Intron sequence **intr_seq**
Output: Ciphertext **ct**

Steps:
Reverse the **pt** and convert into ASCII codes
Convert ASCII codes to binary bits **pt_b**
Reverse the **intr_seq** and convert into ASCII codes
Convert ASCII codes to binary bits **intr_b**
Split **pt_b** by its odd position and even position **pt_b_odd, pt_b_even**
Split **intr_b** by its odd position and even position **intr_b_odd, intr_b_even**

For odd position bits
XOR of **pt_b_odd** and **intr_b_odd**
Get **res_1**
For even position bits
XOR of **pt_b_even** and **intr_b_even**
Get **res_2**
Combine all bits of **res_1** and **res_2** to get **res**
Do the 1-bit left circular rotation on **res**
Convert to **DNAseq,** and then convert to **mRNAseq**
Convert to **tRNAseq**
Map with dynamic DNA encoding table
Get ciphertext **ct**

Journal of Theoretical and Applied Information Technology
31st May 2025. Vol.103. No.10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Finally, the process key has been created by using the Intron sequence and DNA sequence which is given by the sender at an initial time. Process key is having the length of 18 characters and as a first step, reverse the Intron sequence and DNA sequence. The intron sequence is having 2 uppercase characters, 6 digits, 2 lowercase characters. Combine intron and DNA sequence after that add 3 special characters (#, &, +) in the 7, 14, 21st position of the key string and do the 1 –bit left rotation to get the process key.

C. Decryption Method

Once the receiver receives the Ciphertext, we have to do the reverse process to get the plaintext. The receiver generates the DNA encoding table using the process key which is having the Intron sequence and the DNA sequences.

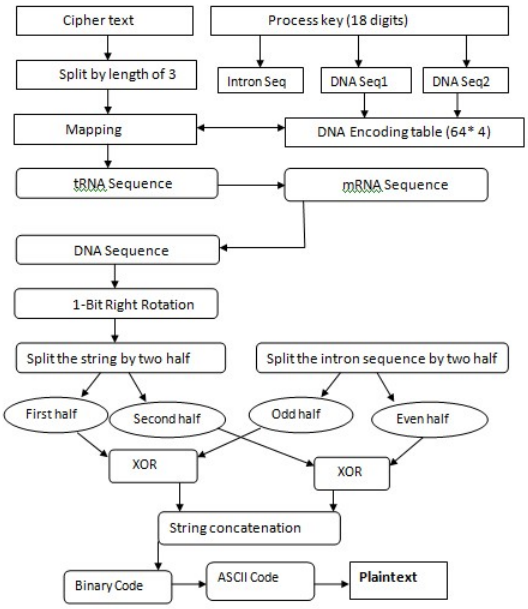

Fig.6. Flow diagram for decryption method

Now, generate the dynamic DNA encoding table using 2 DNA sequences. Split the ciphertext by the length of 3 and map the key codes, fetch the tRNA sequence from the DNA encoding table. Then convert the tRNA sequence to mRNA sequence and then into the DNA sequence. Convert DNA sequence to binary bits and then do 1-bit right rotation on the resultant string. Now, do the XOR operation with the intron sequence and then mix the string by its odd and even position. Finally, reverse the string to get the plaintext.

DNA_Decryption_Method (ct, k)
Input: ciphertext **ct** , key **k**
Output: Plaintext **pt**
Steps:

Retrieve the intron sequence **intron_seq**, DNA sequence 1 **DS_1**, DNA sequence 2 **DS_2**

Generate DNA encoding table **DET**

Split the ciphertext by the length of 3

Map with **DET**

Retrieve **tRNA** sequence

Covert into **mRNA** sequence

Convert into **DNA** sequence

Apply DNA Nucleotide table **DNA_Nuc_Table**

Get the binary y code **bt** of DNA sequence

Do 1-bit right rotation on

Split the string by two halves

First half of string **FH_1**

Second half of the string **SH_2**

Split the intron sequence by two halves
Split the intron sequence by odd position **Odd_str**
Split the intron sequence by odd position **Even_str**

XOR of **FH_1** and **Odd_str**
XOR of **SH_2** and **Even_str**
Concatenate both the strings
Convert into binary string
Convert into ASCII codes
Get the plaintext **pt**

Steps to extract the DNA sequence and Intron sequence

- ✓ Do the 1-bit right rotation on the process key and then remove characters which are in 7, 14 and 21st position of the string.
- ✓ Split the sequence by the length of 10, 4, and 4.
- ✓ Reverse the string individually and get the Intron sequence having a length of 10 and DNA sequences having the length of 4 each.

5. DETAILED EXAMPLE

Below are the inputs given by the sender,
Plaintext: **covid**
Intron sequence: **AB123456cd**
DNA Sequence: **GTAC, TACG**

a) Process key generation:

Take the Intron sequence and 2 DNA sequences. Convert DNA sequence into mRNA and then into tRNA sequence.

Step 1: Convert into mRNA sequence
CATG → CAUG (Replacing Thymine with Uracil)
ATGC → AUGC (Replacing Thymine with Uracil)
Step 2: Convert into tRNA sequence
CAUG → GUAC
AUGC → UACG
Step 3: Concatenate all the strings, and the result is, AB123456cd GUAC UACG
Now, reverse the string individually, and we get below string
dc654321BA CAUG GCAU
Step 4: Now split the string by the length of six and then add three special characters in 7, 14 and 21st position of the string, dc6543# 21BACA& UGGCAU+
Do the 1-bit left circular shift on the string individually, and we get c6543#d 1BACA&2 GGCAU+U. So, final process key is **c6543#d1BACA&2GGCAU+U** and having a length of 21

b) Encryption:

Plaintext: **covid**
Intron sequence: **AB123456cd**
Step 1: Reverse the plaintext and intron sequence. Convert those into ASCII and binary codes. So, the result is **divoc**, **dc654321BA**
Step 2: Binary code of plaintext: **0**1100100 **0**1101001 **0**1110110 **0**1101111 **0**1100011
Binary code of intron sequence: **0**1100100 **0**1100011 **00**110110 **00**110101 **00**110100 **00**110011 **00**110010 **00**110001 **0**1000010 **0**1000001.
Step 3: Now, append 0 to make the plaintext length equal the length of the intron sequence. So, the binary code of the plaintext will be, **00000000 00000000 00000000 00000000 00000000** **0**1100100 **0**1101001 **0**1110110 **0**1101111 **0**1100011.
Step 4: now, split by odd position and even position bits for plaintext and intron sequence
Plaintext: 00000000 00000000 00000000 00000000 00000000 01100100 01101001 01110110 01101111 01100011
Odd position bits: 0000 0000 0000 0000 0000 0100 0110 0101 0111 0101
Even position bits: 0000 0000 0000 0000 0000 1010 1001 1110 1011 1001
Step 5: Intron sequence: 01100100 01100011 00110110 00110101 00110100 00110011 00110010 00110001 01000010 01000001
Odd Bits: 0100 0101 0101 0100 0100 0101 0101 0100 0001 0000
Even Bits: 1010 1001 0110 0111 0110 0101 0100 0101 1000 1001
Step 6: Do the **XOR** operation with odd bits of plaintext and odd bits of intron sequence and then do the same for even bits of both, and the results are,
XOR on odd bits: 0100010101010100010000010011000101100101
XOR on even bits: 1010100101100111011011111101101100110000
Step 7: Concatenate both string and then do 1-bit left shift on the resultant, 0100010101010100010000010011000101100101101010010111001110111011111101101100110000 and the final result is (after 1 – bit left shift) 1000101010101000100000100110001011001011010100101100111011011111101101100110000**0**
Step 8: Convert binary bits to DNA sequence by using a DNA table as shown in TABLE II. So the outcome from the table is GAGGGGGAGAAGCGAGTAGTCCAGTATGTCTTGTCGCGAA
Step 9: Convert DNA sequence into mRNA sequence and the into tRNA sequence. mRNA is, GAGGGGGAGAAGCGAGUAGUCCAGUAUGUCUUGUCGCGAA and the tRNA sequence is CUCCCCCUCUUCGCUCAUCAGGUCAUACAGAACAGCGCUU.
Step 10: Map the tRNA sequence with DNA encoding table and get the key code from dynamic DNA encoding table and the ciphertext is **W$0U^2O$02^2F$02@3C$0B@3 (*10^2**

c) Decryption:

Receiver receivers the ciphertext and the process key from the cloud and the decryption steps are given below.

Ciphertext: **W$0U^2O$02^2F$02@3C$0B@3(*10^2**
Process key: **c6543#d1BACA&2GGCAU+U**

Step 1: To extract the intron sequence and the DNA sequence, split the process key by the length of 7 and then do the 1-bit right circular shift on the process key individually, and the outcome is below.
Split up: c6543#d 1BACA&2 GGCAU+U
1-bit right rotation: **dc6543# 21BACA& UGGCAU+**
Step 2: Remove the characters from the position of 7, 14, 21, and the result is dc6543 21BACA UGGCAU
Step 3: Split the string by the length of 10 (intron) and 4, 4 (2 DNA sequences)
dc654321BA CAUG GCAU
Step 4: Now, reverse the string and we get, **AB123456cd GUAC UACG** and generate the dynamic DNA encoding table using the sequences (GUAC, UACG) and map the key codes as well.
Step 5: Split the ciphertext by the length of 3, map the ciphertext and get the tRNA sequence using dynamic DNA encoding table as shown in TABLE III and the tRNA sequence is, **CUCC CCCU CUUC GCUC AUCA GGUC AUAC AGAA CAGC GCUU**
Step 6: Convert tRNA sequence into mRNA sequence and then into DNA sequence.
mRNA sequence: **GAGG GGGA GAAG CGAG UAGU CCAG UAUG UCUU GUCG CGAA**
DNA sequence: **GAGG GGGA GAAG CGAG TAGT CCAG TATG TCTT GTCG CGAA**
Step 7: Convert the binary bits to DNA sequence using the DNA nucleotide table as shown in TABLE II, and the results are,
10001010 10101000 10000010 01100010 11001011 01010010 11001110 11011111 10110110 01100000
Step 8: Do the 1-bit right circular shift and the outcome is, 01000101 01010100 01000001 00110001 01100101 10101001 01100111 01101111 11011011 00110000. Now split the string by two halves.
String 1: 0100 0101 0101 0100 0100 0001 0011 0001 0110 0101
String 2: 1010 1001 0110 0111 0110 1111 1101 1011 0011 0000
Step 9: Find the binary code of intron sequence:
01100100 **0**1100011 **00**110110 **00**110101 **00**110100 **00**110011 **00**110010 **00**110001 **0**1000010 **0**1000001. Now, append 0 to make the intron sequence equal to the ciphertext length.
Step 10: Split the intron sequence by its odd and even positions
Odd Bits: 0100 0101 0101 0100 0100 0101 0101 0100 0001 0000
Even Bits: 1010 1001 0110 0111 0110 0101 0100 0101 1000 1001
Step 11: Do the XOR operation with the odd position of intron sequence with string1 and do the same using even bits of intron sequence with string 2, and the resultant is,
Result 1 (odd): 0000 0000 0000 0000 0000 0100 0110 0101 0111 0101
Result 2 (even): 0000 0000 0000 0000 0000 1010 1001 1110 1011 1001
Now, mix the bits by placing the bits in odd, even positions, and the final string is,
00000000 00000000 00000000 00000000 00000000 01100100 01101001 01110110 01101111 01100011

Step 12: convert the binary bits to characters (**divoc**) and then reverse the string, so that we can get the plaintext, "**covid**".

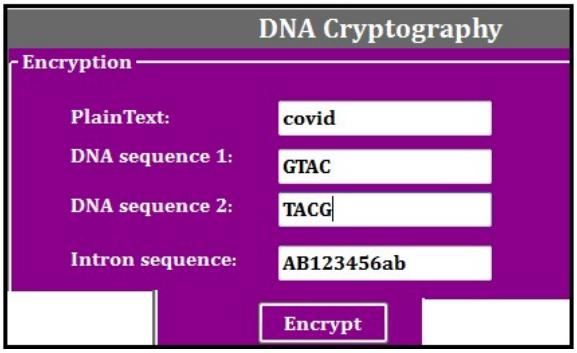

Fig.7. Sample UI design- Encryption

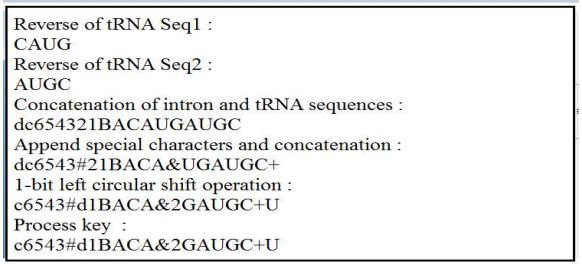

Fig.8. Sample key generation part

Journal of Theoretical and Applied Information Technology
31st May 2025. Vol.103. No.10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

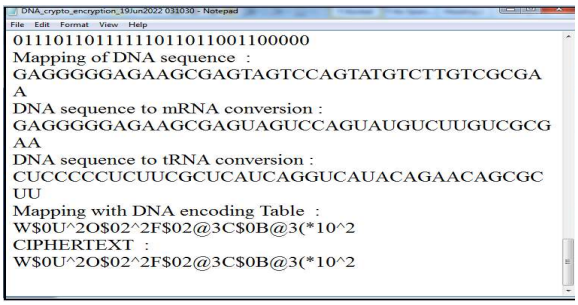

Fig.9.sample ciphertext

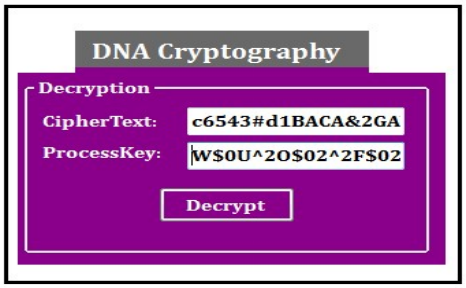

Fig.10. Sample UI design- Decryption

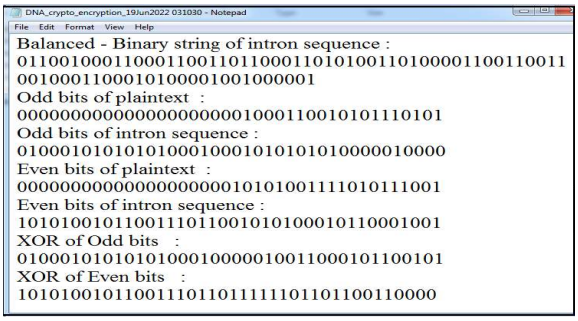

Fig.11. XOR operation of binary bits

Table.3. Dynamic DNA Encoding table

DNA Sequence	Key code	DNA Sequence	Key code	DNA Sequence	Key code	DNA Sequence	Key code
GUUU	0$0	GAUU	0*1	GCUU	0^2	GGUU	0@3
GUUA	1$0	GAUA	1*1	GCUA	1^2	GGUA	1@3
GUUC	2$0	GAUC	2*1	GCUC	2^2	GGUC	2@3
GUUG	3$0	GAUG	3*1	GCUG	3^2	GGUG	3@3
GUAU	4$0	GAAU	4*1	GCAU	4^2	GGAU	4@3
GUAA	5$0	GAAA	5*1	GCAA	5^2	GGAA	5@3
GUAC	6$0	GAAC	6*1	GCAC	6^2	GGAC	6@3
GUAG	7$0	GAAG	7*1	GCAG	7^2	GGAG	7@3
GUCU	8$0	GACU	8*1	GCCU	8^2	GGCU	8@3
GUCA	9$0	GACA	9*1	GCCA	9^2	GGCA	9@3
GUCC	a$0	GACC	a*1	GCCC	a^2	GGCC	a@3
GUCG	b$0	GACG	b*1	GCCG	b^2	GGCG	b@3
GUGU	c$0	GAGU	c*1	GCGU	c^2	GGGU	c@3
GUGA	d$0	GAGA	d*1	GCGA	d^2	GGGA	d@3
GUGC	e$0	GAGC	e*1	GCGC	e^2	GGGC	e@3
GUGG	f$0	GAGG	f*1	GCGG	f^2	GGGG	f@3
UUUU	g$0	UAUU	g*1	UCUU	g^2	UGUU	g@3
UUUA	h$0	UAUA	h*1	UCUA	h^2	UGUA	h@3
UUUC	i$0	UAUC	i*1	UCUC	i^2	UGUC	i@3
UUUG	j$0	UAUG	j*1	UCUG	j^2	UGUG	j@3
UUAU	k$0	UAAU	k*1	UCAU	k^2	UGAU	k@3
UUAA	l$0	UAAA	l*1	UCAA	l^2	UGAA	l@3
UUAC	m$0	UAAC	m*1	UCAC	m^2	UGAC	m@3
UUAG	n$0	UAAG	n*1	UCAG	n^2	UGAG	n@3
UUCU	o$0	UACU	o*1	UCCU	o^2	UGCU	o@3
UUCA	p$0	UACA	p*1	UCCA	p^2	UGCA	p@3
UUCC	q$0	UACC	q*1	UCCC	q^2	UGCC	q@3
UUCG	r$0	UACG	r*1	UCCG	r^2	UGCG	r@3
UUGU	s$0	UAGU	s*1	UCGU	s^2	UGGU	s@3
UUGA	t$0	UAGA	t*1	UCGA	t^2	UGGA	t@3
UUGC	u$0	UAGC	u*1	UCGC	u^2	UGGC	u@3
UUGG	v$0	UAGG	v*1	UCGG	v^2	UGGG	v@3
AUUU	w$0	AAUU	w*1	ACUU	w^2	AGUU	w@3
AUUA	x$0	AAUA	x*1	ACUA	x^2	AGUA	x@3
AUUC	y$0	AAUC	y*1	ACUC	y^2	AGUC	y@3
AUUG	z$0	AAUG	z*1	ACUG	z^2	AGUG	z@3
AUAU	A$0	AAAU	A*1	ACAU	A^2	AGAU	A@3
AUAA	B$0	AAAA	B*1	ACAA	B^2	AGAA	B@3
AUAC	C$0	AAAC	C*1	ACAC	C^2	AGAC	C@3
AUAG	D$0	AAAG	D*1	ACAG	D^2	AGAG	D@3
AUCU	E$0	AACU	E*1	ACCU	E^2	AGCU	E@3
AUCA	F$0	AACA	F*1	ACCA	F^2	AGCA	F@3
AUCC	G$0	AACC	G*1	ACCC	G^2	AGCC	G@3
AUCG	H$0	AACG	H*1	ACCG	H^2	AGCG	H@3
AUGU	I$0	AAGU	I*1	ACGU	I^2	AGGU	I@3
AUGA	J$0	AAGA	J*1	ACGA	J^2	AGGA	J@3
AUGC	K$0	AAGC	K*1	ACGC	K^2	AGGC	K@3
AUGG	L$0	AAGG	L*1	ACGG	L^2	AGGG	L@3
CUUU	M$0	CAUU	M*1	CCUU	M^2	CGUU	M@3
CUUA	N$0	CAUA	N*1	CCUA	N^2	CGUA	N@3
CUUC	O$0	CAUC	O*1	CCUC	O^2	CGUC	O@3
CUUG	P$0	CAUG	P*1	CCUG	P^2	CGUG	P@3
CUAU	Q$0	CAAU	Q*1	CCAU	Q^2	CGAU	Q@3
CUAA	R$0	CAAA	R*1	CCAA	R^2	CGAA	R@3
CUAC	S$0	CAAC	S*1	CCAC	S^2	CGAC	S@3
CUAG	T$0	CAAG	T*1	CCAG	T^2	CGAG	T@3
CUCU	U$0	CACU	U*1	CCCU	U^2	CGCU	U@3
CUCA	V$0	CACA	V*1	CCCA	V^2	CGCA	V@3
CUCC	W$0	CACC	W*1	CCCC	W^2	CGCC	W@3
CUCG	X$0	CACG	X*1	CCCG	X^2	CGCG	X@3
CUGU	Y$0	CAGU	Y*1	CCGU	Y^2	CGGU	Y@3
CUGA	Z$0	CAGA	Z*1	CCGA	Z^2	CGGA	Z@3
CUGC	($0	CAGC	(*1	CCGC	(^2	CGGC	(@3
CUGG	)$0	CAGG	)*1	CCGG	)^2	CGGG	)@3

A DNA encoding table is generated by using two DNA sequences. Initially, a 4 * 4 matrix is formed. After that, it is extended into a 64 * 4

table, which will have 256 characters for key code mapping. The special characters with specified digits are appended at the end of the DNA sequence.

6. PERFORMANCE ANALYSIS

The proposed system had undergone various rounds of testing, and below are the results. The X-axis shows the number of letters; the Y-axis shows the time taken for encrypting and decrypting the data. The graph below shows the number of plaintext characters used for encryption. The corresponding time taken for encryption and decryption is shown below.

Table.4. Sample data analysis on existing work

No of characters	Encryption Time (In Milliseconds)	Decryption Time(In Milliseconds)
4	65	45
8	70	52
16	75	60
32	63	55
64	69	60
128	85	75
256	100	80
512	125	100
1024	350	250

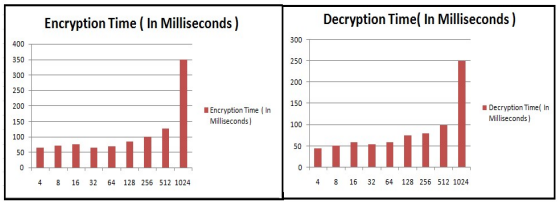

Fig.12.Existing System- Plaintext- Encryption-Decryption

Table.5. Sample data analysis on proposed work

No of characters	Encryption Time (In Milliseconds)	Decryption Time(In Milliseconds)
4	34	31
8	45	34
16	47	36
32	51	41
64	54	43
128	62	56
256	78	64
512	89	80
1024	182	155

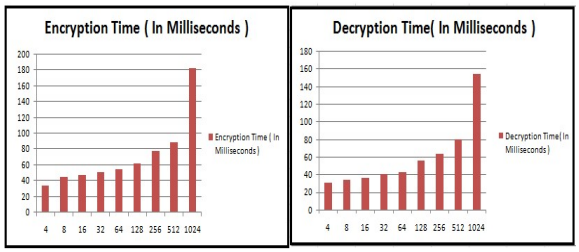

Fig.13.Proposed System-Plaintext- Encryption-Decryption

From the above results, the time complexity is reduced and the robustness of the algorithm are enhanced. Because of the dynamic intron sequence and dynamic encoding table generation, the data cannot be hacked or accessed by any unauthorized user; thus, the implemented DNA cryptosystem is more secure and faster than existing algorithms.

ROBUSTNESS OF THE PROPOSED SYSTEM

The strength of the proposed system had been proven by satisfying the below parameters,

- ✓ Key space analysis
- ✓ Confusion
- ✓ Diffusion

Key space analysis

The key possibilities are very high as the system uses randomness and it's not that easy to find the key (intron sequence) by the intruder. The encoding table can't be hacked easily because the sender chooses the DNA sequence for each and every new transmission in the cloud.

The key also can't be retrieved by any unauthorized person as the key can be sent in encrypted mode instead of sending direct keys while transferring the key in a secure cloud environment. The total probability of finding a key is very high, so the brute force attack is not an easy thing to find a key (Intron sequence) in the proposed system.

Sample key: AB123456yz

Table.6. Key space analysis

Key	Possibility
A	26!
B	26!
1	10!
2	10!
3	10!
4	10!
5	10!
6	10!
y	26!
z	26!

As per above calculation, the total number of key possibilities of the proposed algorithm is,

K = 26! * 26! * 10!* 10! * 10! * 10! * 10! * 10! * 26! * 26!

The security of the proposed scheme is proven from the above evaluation.
K = 26! * 26! * 10!* 10! * 10! * 10! * 10! * 10! * 26! * 26! The result of the above evaluation is, 4.0329146113e+26 *10^60 *26! * 26! * 26! Secret keys are possible (Intron sequence). It is practically impossible to guess the key sequence in an easy manner. So that we can increase the key computational and hacking possibilities while sending information in cloud.

Confusion
If there is any change in the intron sequence, it can change the whole ciphertext as the encoding table is generated based on the intron sequence selection by the sender. Therefore, the 'confusion' property is achieved in the proposed DNA cryptosystem.

Diffusion
If there is any change in the plaintext, the ciphertext will also be changed as we have used the dynamicity property and it is generated dynamically for each and every session initiated by the sender, thereby achieving the diffusion property in the implemented DNA cryptosystem.

If the attacker stores the key or ciphertext from the previous session, and it might be possible to store the key, they will try to do a "dictionary attack" from their point of view to crack the original key. Because we generate intron sequences based on the sender's preferences, we also generate DNA encoding tables based on user input. There is no chance of using the common key or keys with repetitive characters not allowed to be used in the proposed algorithm. To ensure the confidentiality of the data, a set of criteria are defined when choosing the key block. Hence, the proposed algorithm is implemented in a secure way, and finding a key is very difficult.

7. CONCLUSION

In this paper, we have proposed a new modernistic DNA cryptography algorithm to enhance the confidentiality of the data. The result of this approach is that it provides a different and a new ciphertext for the same plaintext at each time, as it's having dynamic logic while creating the DNA encoding table. Moreover, the randomness and dynamic encoding table generation process increase the strength of the encryption algorithm. We have implemented this algorithm using c# (windows form application) language and unit tested with different sizes of data. This scheme is more secure and fast, efficient than the existing algorithm. It provides new confidence for unbreakable algorithms in cryptography science. In future work, we plan to apply this algorithm in cloud data protection domain with the services of e-healthcare, e-commerce, Etc.

REFERENCES

[1] Prema T. Akkasaligar & Sumangala Biradar," Selective medical image encryption using DNA cryptography", 2020, Information Security Journal: A Global Perspective,29:2,91-101,DOI: 10.1080/19393555.2020.1718248
[2] Prasanna Balaji Narasingapuram, M. Ponnavaikko, "DNA Cryptography Based User Level Security for Cloud Computing and Applications ", 2020, International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-5
[3] S. Pratap Singh, M. Ekambaram Naidu, "DNA QR coding for data security using DNA sequence", 2020, Springer, https://doi.org/10.1007/s41870-020-00420-0
[4] A.Vikram, S.Kalaivani ,G.Gopinath ,"A Novel Encryption Algorithm based on DNA Cryptography",2019, Fourth International Conference on Communication and Electronics Systems (ICCES 2019) IEEE

Conference Record # 45898; IEEE Xplore ISBN: 978-1-7281-1261-9.

[5] Anupam Das, Shikhar Kumar Sarma, Shrutimala Deka, "Data Security with DNA Cryptography", the World Congress on Engineering 2019 WCE 2019, July 3-5, 2019, ISBN: 978-988-14048-6-2

[6] Md. Rafiul Biswas, Kazi Md. Rokibul Alam , Shinsuke Tamura , Yasuhiko Morimoto , "A technique for DNA cryptography based on dynamic mechanisms", 2019, Elsevier, Journal of Information Security and Applications 48 (2019)102363,https://doi.org/10.1016/j.jisa.2019.102363 2214-2126

[7] M. Thangavel , P. Varalakshmi , R. Sindhuja, and S. Sridhar ,"Towards Secure DNA Based Cryptosystem", Springer Nature Singapore Pte Ltd. 2018, pp. 163–177, https://doi.org/10.1007/978-981-10-8603-8_14

[8] Nagaraj S M, Mr. S Lokesh,"DNA Cryptography using randomly generated DNA sequence table", 2018 , International Journal of Scientific Development and Research (IJSDR), ISSN: 2455-2631, Volume 3, Issue 5

[9] Animesh Hazra, Soumya Ghosh, and Sampad Jash, "A Review on DNA Based Cryptographic Techniques", International Journal of Network Security, Vol.20, No.6, PP.1093-1104, Nov. 2018 (DOI: 10.6633/IJNS.201811 20(6).10) 1093

[10] M. Thangavel, P. Varalakshmi, "Enhanced DNA and ElGamal cryptosystem for secure data storage and retrieval in cloud", 2017, Springer Science+Business Media, LLC, part of Springer Nature 2017, https://doi.org/10.1007/s10586-017-1368-4

[11] Sajisha K S, Dr. Sheena Math,"An Encryption based on DNA cryptography and Steganography", International Conference on Electronics, Communication and Aerospace Technology ICECA 2017

[12] S.V.Keerthana Priya, S.J.Saritha, "A Robust Technique to Generate Unique Code DNA Sequence", 2017, International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS-2017)

[13] Dr. C. Naga Raju, D Praveen Kumar, D.Hussenappa, V. Kulakarne, "Fast Three Level DNA Cryptographic Technique to Provide Better Security", 2016, DOI: http://dx.doi.org/10.1145/2979779.2979792.

[14] M. Thangavel, P. Varalakshmi, R. Sindhuja, "A Comparative study on DNA Cryptosystem", 2016 FIFTH INTERNATIONAL CONFERENCE ON RECENT TRENDS IN INFORMATION TECHNOLOGY.

[15] Noorul Hussain UbaidurRahman, Chithralekha Balamurugan, Rajapandian Mariappan, "A Novel DNA Computing based Encryption and Decryption Algorithm", International Conference on Information and Communication Technologies,2015, doi: 10.1016/j.procs.2015.02.045.

[16] Wassim Itani , Ayman Kayssi , Ali Chehab , 2009, "Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures" , DOI 10.1109/DASC.2009.139

[17] O. Tornea , M.E. Borda , 2009, "DNA Cryptographic Algorithms", 26, pp. 223–226, 2009

[18] Prajapati Ashishkumar B. Prajapati Barkha, 2016, "DNA Encryption Algorithms: Scope and Challenges in Symmetric Key Cryptography"

[19] Sanchita Paul*, Tausif Anwar , Abhishek Kumar, 2016,"An innovative DNA cryptography technique for secure data transmission" , Int. J. Bioinformatics Research and Applications, Vol. 12, No. 3, 2016

[20] Bismi Beegom S, Dr. Sangeetha Jose, ICECA 2017,"*An Enhanced Cryptographic Model Based on DNA Approach" ,* 978-1-5090-5686-6/17/

[21] Ramya Princess Mary , P. Eswaran , K.Shankar, "Multi Secret Image Sharing Scheme based on DNA Cryptography with XOR", Volume 118 No. 7 2018, 393-398 ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)

[22] Sudipta Singha Roy, Shaikh Akib Shahriyar, Md. Asaf-Uddowla, Kazi Md. Rokibul Alam, Yasuhiko Morimoto , 2017 , 978-1-5386-1150-0/17/ ,"A Novel Encryption Model for Text Messages using Delayed Chaotic Neural Network and DNA Cryptography"

[23] Hamza Hammami , Hanen Brahmi , Sadok Ben Yahia, 2018, 978-1-5386-2290-2/18 ,"Secured Outsourcing Towards a Cloud Computing Environment Based on DNA Cryptography"

[24] Animesh Hazra, Soumya Ghosh, and Sampad Jash, 2018 , "A Review on DNA Based Cryptographic Techniques", International Journal of Network Security, Vol.20, No.6, PP.1093-1104, Nov. 2018 (DOI: 10.6633/IJNS.201811 20(6).10) 1093

[25] Nabarun Nandy , Debanjan Banerjee, Chittaranjan Pradhan "Color image encryption using DNA based cryptography" , 2017 , https://doi.org/10.1007/s41870-018-0100-9