International Journal of Engineering Technology and Applied Science

(ISSN: 2395 3853), Vol. 9 Issue 11 October 2023

Implementation of Two-Fold DNA Cryptography Based on Amino Acid Table in Cloud Computing

Manmohan Singh[1]
[2]Department of Computer Science and Engineering, IES College of Technology, Bhopal, India
Dheresh Soni[2]
School of Computing Science and Engineering, VIT Bhopal University, Bhopal, India

Abstract: Worldwide, organizations are migrating their IT infrastructure to public clouds. One of the biggest challenges during this transition is the implementation of appropriate security techniques to withstand cyber attacks. Unfortunately, this process is still ambiguous for many organizations and data are exposed to different threats and vulnerabilities during this migration. To address this, there are several approaches applied worldwide to strengthen security of the stored data on cloud computing. The two-fold DNA Cryptography is one such data security techniques. The existing technique focuses only on the ASCII character set, ignoring the non-English user of the cloud computing. Thus, proposing two-fold DNA Cryptography using the Unicode characters so that it can be used for non-English users also.

Key Word: Cryptography, Cyber-attacks, Public clouds, Data security, cloud computing

I. Introduction

The major concern of cloud security is to mostly non-English users, because existing techniques use the ASCII character set, serving to the English-speaking users only. Unlike traditional cryptography, DNA cryptography depends on DNA characteristics along with crypto-graphic techniques to ensure security. Valuable properties of this technique are: self-assembling criteria of DNA molecules, parallel computations, large data storage capacity, etc. Traditional crypto algorithms are based on mathematical techniques that weaken the robustness of their encryption process. For example, recent studies show that like AES, MD5, RSA, DES, etc are not secure enough as in [3] and [4]. It is expected that by proposing unbreakable crypto-system, DNA cryptography would ensure the data security for the next generation.

The DNA cryptography combines the mathematical computation and biological computation. The proposed system converts plaintext to Unicode, then to Hexadecimal, and then to Binary form. The binary code is XO-Red with a private key to get an updated Binary code. This is called XOR Cipher. Then the DNA sequences are generated using a DNA Sequence Table. The DNA sequence is processed through some biological mechanisms like mRNA, t-RNA and reverse simulation as in [5]. Finally amino acid table consists of protein sequences grouped according to 26 ASCII capital letters is used to get the final compressed cipher-text. Thus the two fold security, that is, both mathematical and biological computation disable intruders to find out the correct sequence of plaintext.

II. Literature Survey

Prajapati Ashokkumar B Et.al in [2] has proposed an approach which is based on two-serial DNA encryption algorithm which provides two-layer protection in cloud computing. The proposed algorithm also uses the Unicode character set that will be helpful for encoding non-English languages. It follows symmetric cryptography.

Now days, the field of biology and that of cryptography have come to combine. The study of DNA can be applied in DNA cryptography systems that are based on DNA and onetime-pads, and if it is used correctly, it is virtually impossible to crack the system [4]. The size of one-time-pad depends on the cryptographic system. There are various procedures for DNA one-time-pad encryption schemes [5]. Currently DNA technology is based upon the modern biological technologies which are extensively laboratory dependent. There is not any specific general theory about applying DNA molecules into cryptography [6].

DNA cryptographic technique based on dynamic DNA sequence table along with OTP. Multiple steps of DNA conversions and DNA sequences increase the secrecy of cipher-r-text in [3].

III. Proposed System

The DNA cryptography algorithm provides the two level of security. The proposed system focuses on extending the DNA cryptography algorithm to be used with Unicode character set along with both mathematical and biological computation as in [3].

DNA cryptography is still an emerging field of research. The technique proposed considers a biological simulation technique that is based on DNA encoding table along with the mathematical computation. The mathematical computation involved is the conversion of the plaintext to Unicode, Unicode to Hexadecimal, and finally Hexadecimal to Binary. In information science, the binary digital coding encoded by two state 0 or 1 and a combination of 0 and 1. But DNA digital coding can be encoded by four kind of base, that is ADENINE (A) and THYMINE (T) or CYTOSINE (C) and GUANINE (G) as in [2]. There are possibly 4! = 24 pattern by encoding format like (0123/ATGC). The DNA Encoding is done using the Table 1, where every two-bit binary is replaced with it's equivalent DNA representation.

Table 1 DNA Encoding

DNA	Digital Code
00	A
01	T
10	G
11	C

The biological DNA replication to RNA i.e. mRNA(replace T with U), tRNA(replace A with U, U with A, C with G and G with C), reverse simulation process(replace U with T), and conversion of RNA to amino acid table to get cipher-text are used to folding up the message into multiple times. Conversion of RNA to amino acid uses protein sequence from Table 2 where the protein sequence is replaced by a capitalized letter i.e. 'A' for "GCT". Amino acid has multiple cordons. Therefore, a number of DNA cordons may present in the same group of protein sequence as in [3]. According to the table 2, 26 letters are mapped to protein sequences. As the mapping process is random, it also increases the secrecy.

Table 2 Single Letter/Alphabet mapping to protein Sequence

Alphabets/ letters	Protein Sequence
A	GCT, GCC, GCA, GCG
B	TAA, TAG
C	TGT, TGC
D	GAT, GAC
E	GAA, GAG
F	TTT, TTC
G	GGT, GGC, GGA, GGG
H	CAT, CAC
I	ATT, ATC, ATA
J	TGA
K	AAA, AAG
L	CTT, CTC, CTA, CTG
M	ATG
N	AAT, AAC
O	TTA, TTG
P	CCT, CCC, CCA, CCG
Q	CAA, CAG
R	CGT, CGC,CGA, CGG
S	TCT, TCC, TCA, TCG
T	ACT, ACC, ACA, ACG
U	AGA, AGG
V	GTT, GTC, GTA, GTG
W	TGG
X	AGT, AGC
Y	TAT
Z	TAC

Algorithm of Encryption

Step 1: Convert Original Plaintext (non-English also) to uni-code sequence.
Step 2: Convert generated uni-code to Hexadecimal and then obtained Hexadecimal to Binary sequence.
Step 3: Randomally generate private key and XOR with generated Binary sequence to get updated Binary sequence.
Step 4: Convert updated Binary Sequence to DNA sequence using DNA Sequence Table as mentioned in Table 1.
Step 5: Generate mRNA Sequence by replacing The Thymine (T) is replaced with Uracil (U).
Step 6: Generate tRNA by replacing A→U, U→A, G→C and C→G.
Step 7: tRNA is divided into two parts (1st and 2nd). Generate updated tRNA Sequence by shuffling 1st part and 2nd part.
Step 8: Generate Reverse Simulation of tRNA by replacing Uracil (U) with Thymine (T).
Step 9: Apply Amino Acid table on generated Sequence as mentioned in Table 2 and Finally generate Ciphertext.

Algorithm of Decryption

Step 1: Get Ciphertext and using Amino Acid table for each matching protein sequence, generate simulated tRNA sequence.
Step 2: Replace Thymine (T) with Uracil (U) to get the tRNA Sequence.

Step 3: Generate original tRNA Sequence by shuffling 1st part and 2nd part of tRNA sequence.
Step 4: Generate mRNA by replacing U→A, A→U, C→G and G→C.
Step 5: Generate DNA Sequence by replacing Uracil (U) is replaced with Thymine (T).
Step 6: Generate Binary Sequence from DNA sequence using DNA Sequence Table.
Step 7: XOR with Binary sequence with private key to get the original Binary sequence.
Step 8: Convert obtained Binary to Hexadecimal and obtained Hexadecimal to Unicode sequence.
Step 9: Finally convert Unicode to original plaintext.

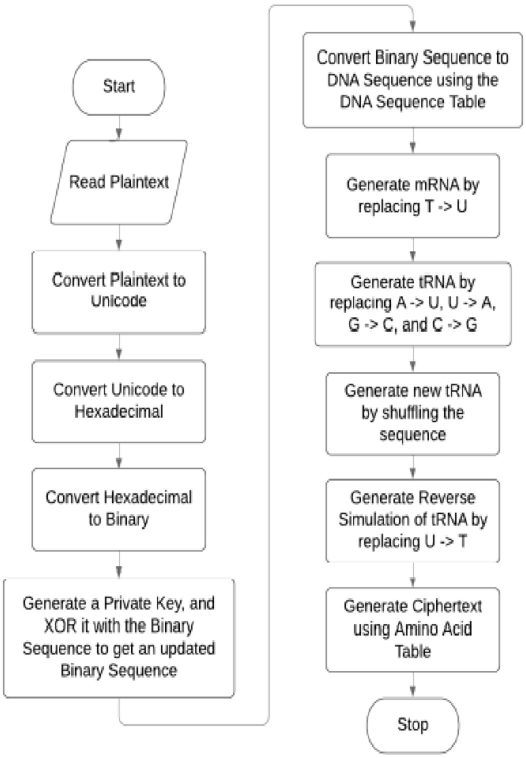

Figure 1 Flowchart for the Encryption Process.

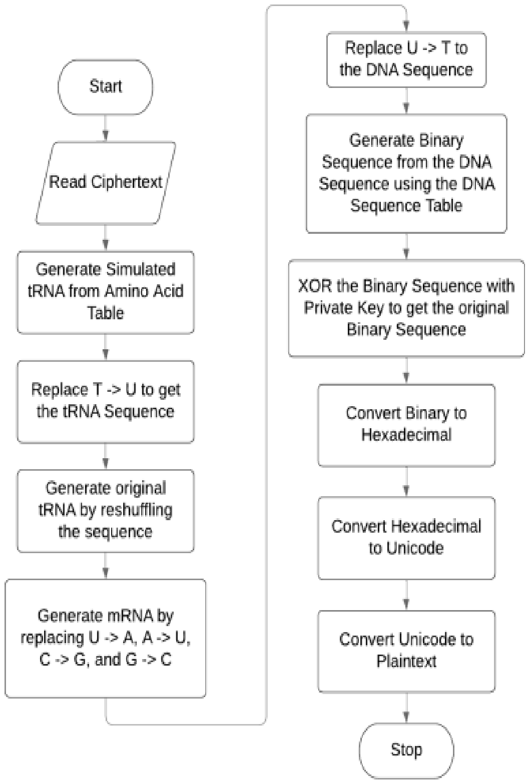

Figure 2: Flowchart for the Decryption Process.

IV. Experimental Analysis

4.1. Experimental Setup

The prototype of the proposed technique is developed under the environment on Intel(R) Core-TM i5-2430M 2.50 GHz 64 bit processor with 8 GBytes of RAM running on Windows 10 operating system. It is developed in Java employing Eclipse Mars 1.0 along with jdk1.8.1 as kit where IDE default storage is used for storing data. Byte size matters calculator is used to measure the size of text.

Encryption Example - Output of Encryption process using plaintext ’Testcase’.

Plaintext : Testcase

Unicode - \u0054\u0065\u0073\u0074\u0063\u0061\u0073\u0065

Hexadecimal - 45c75303035345c75303036355c75303037335c753030373
45c75303036335c75303036315c75303037335c7530303635

Binary
1010001011100011101010011000000110000000110110
0011010101011100011101010011000000110000000011
0111001100110101110001110101001100000000110000
0011011100110100010111000111010100110000000011
0000001101100011001101011100011101010100110000
0011000000110110001100010101110001110101010011
0000001100000001101110011001101011100011101010 1
001100000001100000001101100011010 1

Decrypting-Key

Updated-Binary
010001011100011101010011000000110000000110110
0011010101011100011101010011000000110000000011
0111001100110101110001110101001100000000110000
0011011100110100010111000111010100110000000011
0000001101100011001101011100011101010100110000
0011000000110110001100010101110001110101010011
0000001100010011010100111010001011011000 1001
01100111000110111000111110011011

DNA-Digital-Code
TTCATCTTACAAACAAACTTACTATTCATCTTACA
AACAAACTGACTTTTCATCTTACAAACAAACTCA
CCTTCATCTTACAAACAAACTCACTATTCATCTTA
CAAACAAACTGACACTTCATCTTACAAACAAACT
GACATTTCATCTTACAAACATACTTACGGAGCTG
AGTTGTCATGCGACCGTGC

mRNA-Code
UUCAUCUUACAAACAAACUUACUAUUCAUCUUA
CAAACAAACUGACUUUUCAUCUUACAAACAAAC
UCACACUUCAUCUUACAAACAAACUCACUAUUC
AUCUUACAAACAAACUGACACUUCAUCUUACAA
ACAAACUGACAUUUCAUCUUACAAACAUACUUA
CGGAGCUGAGUUGUCAUGCGACCGUGC

tRNA-Code
AAGUAGAAUGUUUGUUUGAAUGAUAAGUAGAAU
GUUUGUUUGACUGAAAAGUAGAAUGUUUGUUUG
AGUGUGAAGUAGAAUGUUUGUUUGAGUGAUAAG
UAGAAUGUUUGUUUGACUGUGAAGUAGAAUGUU
UGUUUGACUGUAAAGUAGAAUGUUUGUAUGAAU

GCCUCGACUCAACAGUACGCUGGCACG

Updated-tRNA-Code
AGAAUUUUUAUAAGAAUUUUUAUAAGAAUUUUU
AUUAGAAUUUUUAUAAGAAUUUUUAUUAGAAUU
UUUAUUAGAAUUUUUAUCUGCCAAUCCGCCAUG
AGUGUGAGUAUGAGUGUGCGAAUGAGUGUGGGG
AUGAGUGUGGGUAUGAGUGUGCGGAUGAGUGUG
CGAAUGAGUGAGAGCCAUACGAGUGAG

Reverse-Simulated-Code
AGAATTTTTATAAGAATTTTTATAAGAATTTTTATT
AGAATTTTTATAAGAATTTTTATTAGAATTTTTATT
AGAATTTTTATCTGCCAATCCGCCATGAGTGTGA
GTATGAGTGTGCGAATGAGTGTGGGGATGAGTG
TGGGTATGAGTGTGCGGATGAGTGTGCGAATGA
GTGAGAGCCATACGAGTGAG

Ciphertext
UIFIUIFIUIFIUIFIUIFIUIFIUIFICOSAMXVXMX
VRMXVGMXVGMXVRMXVRMXEXHTXE00020
0020000000200000000000110110030003200330
03000330032001103O1

Decryption Example - Example to get back the Plaintext from Ciphertext.

Ciphertext
UIFIUIFIUIFIUIFIUIFIUIFIUIFICQSAMXVXMX
VRMXVGMXVGMXVRMXVRMXEXHTXE00020
0020000000200000000000110110030003200330
0300033003200110301

Decompressed-code
AGAATTTTTATAAGAATTTTTATAAGAATTTTTA
TTAGAATTTTTATAAGAATTTTTATTAGAATTTTT
ATTAGAATTTTTATCTGCCAATCCGCCATGAGTG
TGAGTATGAGTGTGCGAATGAGTGTGGGGATGA
GTGTGGGTATGAGTGTGCGGATGAGTGTGCGAA
TGAGTGAGAGCCATACGAGTGAG

Reverse-Simulated-code
AGAAUUUUUAUAAGAAUUUUUAUAAGAAUUUUU
UAUUAGAAUUUUUAUAAGAAUUUUUAUUAGAA
UUUUUAUUAGAAUUUUUAUCUGCCAAUCCGCC
AUGAGUGUGAGUAUGAGUGUGCGAAUGAGUGU
GGGGAUGAGUGUGGGUAUGAGUGUGCGGAUGA
GUGUGCGAAUGAGUGAGAGCCAUACGAGUGAG

Rearranged-code

AAGUAGAAUGUUUGUUUGAAUGAUAAGUAGAA
UGUUUGUUUGACUGAAAAGUAGAAUGUUUGUU
UGAGUGUGAAGUAGAAUGUUUGUUUGAGUGAU
AAGUAGAAUGUUUGUUUGACUGUGAAGUAGAA
UGUUUGUUUGACUGUAAAGUAGAAUGUUUGUA
UGAAUGCCUCGACUCAACAGUACGCUGGCACG

tRNA-Code

OUCAUCUUACAAACAAACUUACUAUUCAUCUU
ACAAACAAACUGACUUUUCAUCUUACAAACAA
ACUCACACUUCAUCUUACAAACAAACUCACUA
UUCAUCUUACAAACAAACUGACACUUCAUCUU
ACAAACAAACUGACAUUUCAUCUUACAAACAU
ACUUACGGAGCUGAGUUGUCAUGCGACCGUGC

Reverse-Simulated-code

TTCATCTTACAAACAAACTTACTATTCATCTTA
CAAACAAACTGACTTTTCATCTTACAAACAAAC
TCACACTTCATCTTACAAACAAACTCACTATTC
ATCTTACAAACAAACTGACACTTCATCTTACAA
ACAAACTGACATTTCATCTTACAAACATACTTA
CGGAGCTGAGTTGTCATGCGACCGTGC

Binary-code

1010001011100011101010011000000110000001101
1000110101010111000111010100110000001100000
0110111001100110101110001110101001100000011
0000001101110011010001011100011101010011000
0001100000011011000110011010111000111010100
1100000011000000110110001100010101110001110
1010011000000110001001101010011101000101101
10001001011001110001101110001111110011011

Original-Binary-Code

1010001011100011101010011000000110000001101
1000110101010111000111010100110000001100000
0110111001100110101110001110101001100000011
0000001101110011010001011100011101010011000
0001100000011011000110011010111000111010100
1100000011000000110110001100010101110001110
1010011000000110000001101110011001101011100
01110101001100000011000000110110001101101

Hexadecimal-code

45c75303035345c75303036355c75303037335c75303037345c75303036335c75303036315c75303037335c7530303635

Unicode - \u0054\u0065\u0073\u0074\u0063\u0061\u0073\u0065

Original-Code : Test-case

4.2. Experimental Setup Snaps

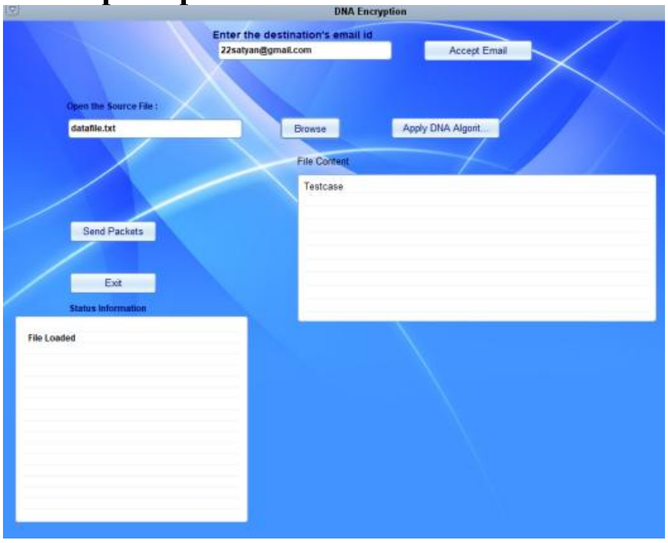

Figure 3 DNA Encryption Module after entering the receiver's email and the file to encrypt.

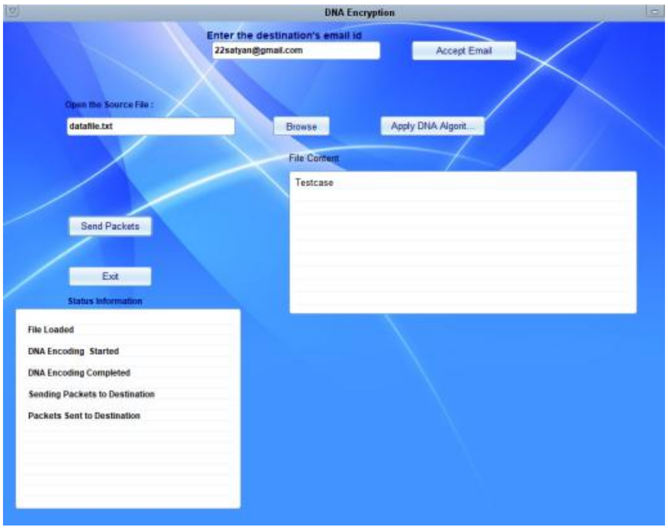

Figure 4 DNA Encryption Module after performing the encryption process.

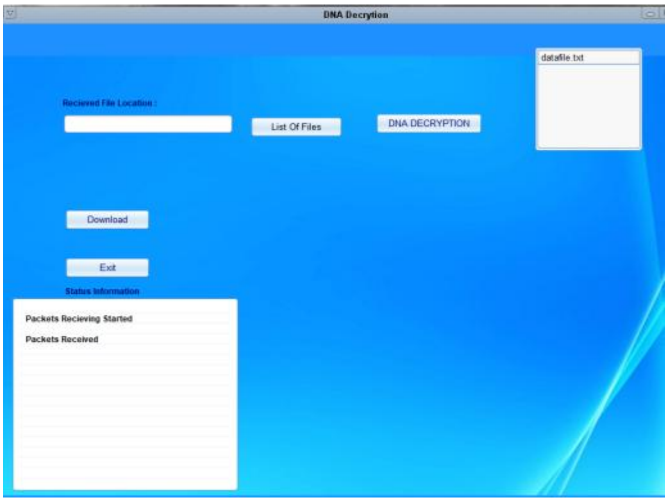

Figure 5 DNA Decryption Module after receiving the packets.

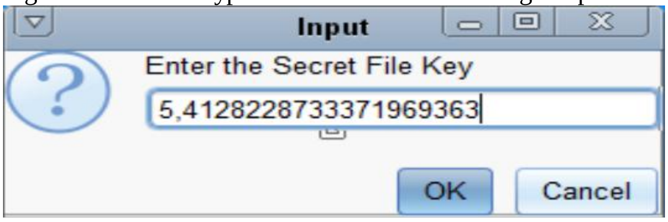

Figure 6 Pop-up to enter the Secret File Key.

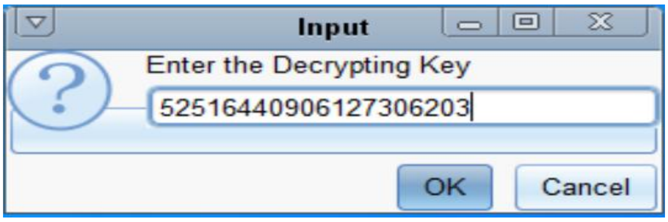

Figure 7 Pop-up to enter the Decrypting Key.

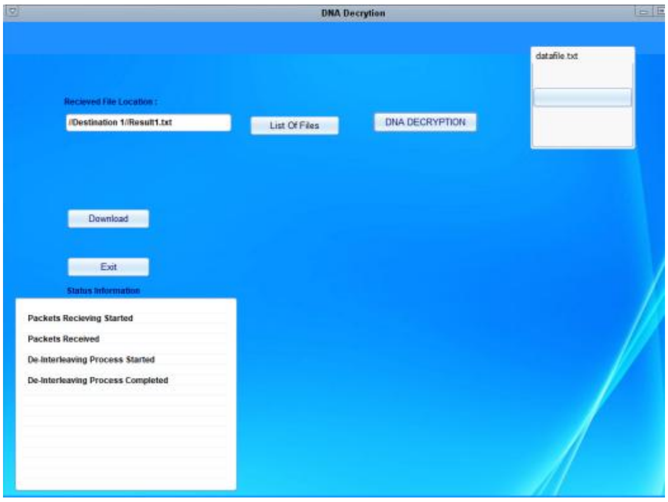

Figure 8 DNA Decryption Module after performing the decryption process.

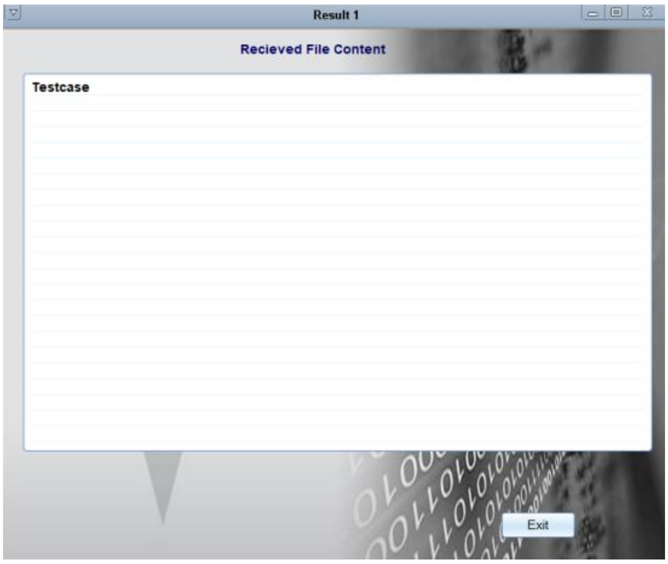

Figure 9 Decrypted Content of the file.

V. Experimental Results

The security of the encryption process come from two levels - first, the random key generated for XOR Cipher, and, second, the key for accessing the file generated using the RSA Algorithm. The data can only be accessed and decrypted only if both the keys are present. The execution time for encoding-decoding process and the encryption-decryption procree is very less and almost similar.

VI. Conclusion

Data security is the main challenge for cloud usability. Various algorithms like RSA, Diffie-Hellman, DNA encryption etc. are available to provide data security for the data stored on cloud. Digital signatures, Extensible Authentication Protocols are used for authentications. But, most of these existing algorithms only utilize the concepts of mathematical computations. Also, we have seen that almost all of the existing algorithms use only the ASCII Character Set, thus focusing on only the English-speaking customers. Using DNA Cryptography Algorithm, we achieve the power of using biological computations as well. The proposed system uses the mathematical and biological computation which disable intruders to find out the correct sequence of plaintext. We have also extended the Cryptography Algorithms to be used with Unicode Character Set so that it can be used by the non-English users as well. This can help reach to the wider community of the cloud users.

Reference:

1. L. U. Mingxin, L. Xuejia, X. Guozhen, Q. Lei, *Symmetric-key cryptosystem with DNA technology*, Science in China Series F: Information Science, Springer Verlag, Germany, vol. 50, no. 3, pp. 325-333, 2007.
2. pajapati Ashishkumar B., Prajapati Barkha, *Implementation of DNA Cryp-tography in Cloud Computing and Using Socket Programming*, 2016 Inter-national Conference on Computer Communication and Informatics (ICCCI-2016), Jan. 07 – 09, 2016, Coimbatore, India.

3. Emtious Md. Sazzad Hossain, Kazi Md. Rokibul Alam, Md. Rafiul Biswas & Yasuhiko Morimoto, *A DNA Cryptographic Technique Based on Dynamic DNA Sequence Table*, 19th International Conference on Computer and Information Technology, December 18-20, 2016, North South University, Dhaka, Bangladesh.
4. PrashantRewagad, YogitaPawar, *Use of Digital Signature with Diffie- Hellman Key Exchange and AES Encryption Algorithm to Enhance Data Security in Cloud Computing*, 2013 International Conference on Communication System and Network Technologies (IEEE Computer Society).