Digital Encoding to the form of Amino Acids for DNA Cryptography and Biological Simulation

Mona Sabry
Computer Science dept.,
Faculty of Computer Science and Information Systems,
Ain Shams University,
Cairo, Egypt.

Mohamed Hashem
Information Systems dept.,
Faculty of Computer Science and Information Systems,
Ain Shams University,
Cairo, Egypt.

Taymoor Nazmy
Computer Science dept.,
Faculty of Computer Science and Information Systems,
Ain Shams University,
Cairo, Egypt.

ABSTRACT

In a way to minimize the gap between digital computing and DNA computing, it is needed to transfer DNA between the two fields, and to make use of the two technologies in generating ideas of data integrity and information security. One of the critical problems in amino acid analysis is how to establish a digital coding system to better reflect the properties of amino acids and their degeneracy. This paper introduces a method to convert digital data to the form of DNA and then to the form of amino acids using the natural RNA codons distribution on the 20 natural amino acids which preserves their biological properties. The Decoding method also convert the amino acids to a digital form. The method solves the problem of ambiguity that more than one codon correspond to the same amino acid. Applicability and reversibility of the method is proven and successfully implemented.

The presented encoding method can serve in DNA computers and biological experiments by representing data in the form of amino acids. This mainly aims at increasing the flexibility of converting data between biological medium and digital medium. Although it does not include the use of secret key but it can also be used as an auxiliary factor in cryptographic and steganographic applications like data integrity and digital signature.

Keywords

Amino acids; binary data; digital encoding; DNA; RNA; cryptography; secret writing; Ambiguity steganography; biological simulation.

1. INTRODUCTION

DNA computing is a branch of computing which uses DNA, biochemistry, and molecular biology hardware, instead of the traditional silicon-based computer technologies. In this trend many researches; including held by Sabry et al. [1]; were directed to find a way to encode binary data to the form of DNA. DNA cryptography is a new born cryptographic field which emerged with the research of DNA computing. The massive parallelism and vast information density inherent in DNA molecules are explored for cryptographic purposes such as encryption, data hiding, signature, and so on [2, 3].

The research in this area has two directions. One direction which is led in biological labs like technologies using Polymerase Chain Reaction (PCR), DNA synthesis, hybridization and DNA digital coding, which have been developed and their results were well accepted [4, 5, 6]. Another direction used the DNA structure to implement cryptographic approaches on digital computers.

Those researches have presented different ways to initially convert data to the form of DNA prior to their cryptographic or steganographic method. In 1999, Clelland et al. [6] presented a steganographic approach in which they hide secret messages encoded as DNA strands among a random DNA strand. In 2000, Prof. Gehani [5] presented DNA-based one-time-pads mechanisms that are used to design two encryption methods. One is called substitution in which is the DNA plain code sequence is translated to DNA cryptograph sequence according to a defined mapping. The other is called Exclusive-OR method, which uses biological molecular techniques to perform an Exclusive-OR operation of DNA plain code and DNA cipher key sequence [5].

Ning Kang then led another approach in [7]. In his research, he did not apply real DNA computing operations, but he just used the principle ideas in central dogma of molecular biology and the DNA structure in order to develop his cryptographic method. The method only simulates the transcription, splicing, and translation processes to develop a cryptographic method. He also hasn't found an efficient solution the ambiguity problem. This problem is introduced and solved in our proposed algorithm. Heider et al. presented a survey on various ways that have been used for data encoding to DNA form [8]. Then they presented their own encoding algorithm as a prior to their introduced cryptographic algorithm.

In a way of concentration on the idea of data encoding, Sabry et al. [1] presented three different methods to convert data from binary form to DNA form and then to amino acids form. These transformation methods were implemented using artificially constructed tables of RNA codons which contained 26 virtual amino acids. In our paper, we will use the natural distribution of RNA codons on the 20 natural amino acids for the same purpose.

The importance of such transformation lies mainly in representing data in a biological form that can make data able to go through biological experiments and processes, especially related to Amino Acids and DNA. It is also a way of viewing data moving through biological processes and representing it in a binary form which can be used in many computer applications.

In the field of cryptography, the encoding techniques cannot provide security by their own as they don't include the use of a secret key. But they can be embedded into another encryption algorithm to enhance confusion and therefore enhance security. This concept is suitable for applying data integrity, digital signature and confidentiality. An encryption algorithm is presented in [9] which uses the idea of converting digital data to the form of amino acids using secret key.

The next sections are organized as follows: section 2 explains biological background information that helps us understand the biological concepts involved in our algorithm. Section 3 contains the standard codons table design and how it will be used then the details of our encoding algorithm, its reverse and complexity calculation. Section 4 involves discussions and analysis about the algorithms and their applications. Section 5 presents our conclusion.

2. BIOLOGICAL BACKGROUND

DNA is made up of two base pairs (strands) [10]. The base pairs are formed of four bases (nucleotides) that repeat over and over in a very long molecule. DNA strands are long polymers that consist of millions of linked nucleotides [10]. The nucleotides that make up these polymers are named after the nitrogen base that it consists of. The nucleotides are: **Adenine (A), Cytosine (C), Guanine (G) and Thymine (T) [10].**

The first DNA operation is Transcription. The DNA code is transcribed to RNA code, which is still in the form of nitrogenous bases, except for Thymine on the DNA pairs which is replaced with uracil 'U' on the RNA code. The next operation is Translation. The RNA code is translated to protein code, which is a different form. Each amino acid is formed by combining three bases on the RNA. These three-nucleotide sequences on the mRNA are called codons. Each codon corresponds to a specific amino acid. One or more codons can correspond to the same amino acid. The amino acids are then organized into the correct sequence to build a protein. Figure 1 graphically shows these steps.

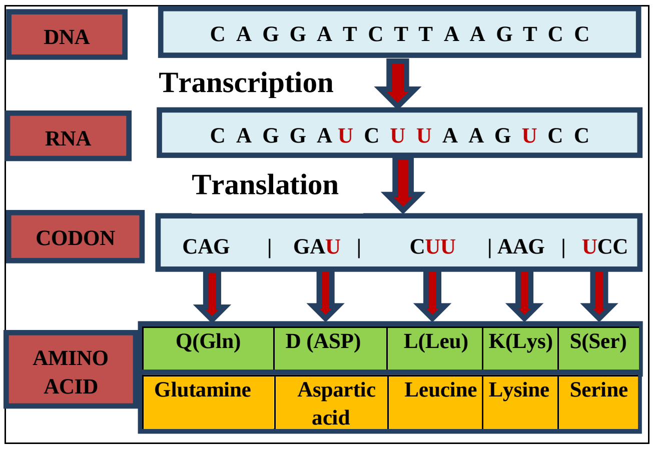


Fig 1: Illustration with example to DNA conversion through processes of transcription and translation.

3. OUR PROPOSED METHODOLOGY

3.1 The Encoding to DNA and the Distribution Tables

Data in binary form can be transferred to DNA or RNA form using Table 1 which is direct representation of each 2 bits to a single DNA character. Note that the only difference between DNA and RNA is that letter 'T' in DNA is the same as letter 'U' in RNA. The RNA code can be transferred to the Amino acids' form using Table 2. This table is the standard universal table of Amino acids and their corresponding codons' representation in the form of RNA. Each amino acid has a name, abbreviation, and a single character symbol (1-letter form). This character symbol is what will be used in our algorithm.

Table 1. Bits encoding to the forms of DNA and RNA.

Bit 1	Bit 2	RNA	DNA
0	0	A	A
0	**1**	**C**	**C**
1	**0**	**G**	**G**
1	**1**	**U**	**T**

The encoding method will depend on Table 2 which contains the natural distribution of different codons on the 20 natural amino acids. We will also consider the stop codons as an additional amino acid which we will call 'B' resulting in 21 amino acids. As each amino acid is made up of 3 RNA, all the 64 combinations of the 3 RNA codons are distributed on the 21 amino acids. The new distribution is presented in Table 3. The direct way to convert any binary data is:

1- Convert from binary to DNA: Using Table 1 which contains direct one-to-one transformation of different combinations of 2 bits to the one of the four DNA [A,C,G,T]or RNA [A,C,G,U] bases.

2- Convert from DNA to amino acids: Using Table 3, take each 3 RNA to be named as one codon. By searching table 3 to find the codon location, and substitute the codon with the column name (the amino acid character).

For example "010100111010110100" is encoded to "CCA UGG UCA" in the form of RNA. It is encoded to "PWS" in the form of amino acids. This way seems so simple and represents the natural amino acid form of the data. But, it suffers a drawback while decoding back the data from amino acid's form to binary form.

Given one amino acid character, it is needed to search Table 3 to find its column. On finding its column, we cannot just substitute it with a DNA codon, because the amino acid may contain more than one codon and there is no available way to know which one was the actual or initial codon. This amino acids' property is what is called "Ambiguity".

The ambiguity problem is the problem of codon-amino acid mapping which aroused with other algorithms based on the concept of Central Dogma. One of the researchers who faced this problem is Kang Ning in [7]. Ning handled this problem by putting this codon-amino acid mapping in the secret key. Then this key is to be sent to the receiver through a secure channel [7]. This idea is not efficient because the key size increases in relation to the plaintext size. But the problem is handled in this research in a different way.

The information about the right ambiguity should be included during the encoding process to select which codon is the right representation of the amino acid. The ambiguity within each amino acid varies between the numbers of (1, 2, 3, 4 and 6). For representing these numbers, they need different number of bits for each. The ambiguity of each amino acid and the needed number of bits are mapped in Table 3.

In some cases all bits combinations will cover the number of ambiguities and other cases not, like if ambiguity =1 or 3 or 6. For example on having six ambiguities which need 3 bits for representation, the bits should define the ambiguity values (0, 1, 2, 3, 4, and 5), while the total combinations of 3 bits represent the values (0, 1, 2, 3, 4, 5, 6, and 7). So the values (6 and 7) or (110, 111) do not represent a real ambiguity value and will not be used.

Table 2. The RNA codon table [https://en.wikipedia.org/wiki/Genetic_code]

Nonpolar	Polar	Basic	Acidic	(stop codon)

Standard genetic code									
1st	2nd base								3rd
Base	U		C		A		G		Base
U	UUU	(Phe/F) Phenylalanine	UCU	(Ser/S) Serine	UAU	(Tyr/Y) Tyrosine	UGU	(Cys/C) Cysteine	U
	UUC		UCC		UAC		UGC		C
	UUA	(Leu/L) Leucine	UCA		UAA	Stop (Ochre)	UGA	Stop (*Opal*)	A
	UUG		UCG		UAG	Stop (*Amber*)	UGG	(Trp/W) Tryptophan	G
C	CUU		CCU	(Pro/P) Proline	CAU	(His/H) Histidine	CGU	(Arg/R) Arginine	U
	CUC		CCC		CAC		CGC		C
	CUA		CCA		CAA	(Gln/Q) Glutamine	CGA		A
	CUG		CCG		CAG		CGG		G
A	AUU	(Ile/I) Isoleucine	ACU	(Thr/T) Threonine	AAU	(Asn/N) Asparagine	AGU	(Ser/S) Serine	U
	AUC		ACC		AAC		AGC		C
	AUA		ACA		AAA	(Lys/K) Lysine	AGA	(Arg/R) Arginine	A
	AUG	(Met/M) Methionine	ACG		AAG		AGG		G
G	GUU	(Val/V) Valine	GCU	(Ala/A) Alanine	GAU	(Asp/D) Aspartic acid	GGU	(Gly/G) Glycine	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	(Glu/E) Glutamic acid	GGA		A
	GUG		GCG		GAG		GGG		G

Table 3. Another layout to RNA codon table

Amino acid	A	B	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
#Ambiguity	4	3	2	2	2	2	4	2	3	2	6	1	2	4	2	6	6	4	4	1	2
0	GCU	UAA	UGU	GAU	GAA	UUU	GGU	CAU	AUU	AAA	UUA	AUG	AAU	CCU	CAA	CGU	UCU	ACU	GUU	UGG	UAU
1	GCC	UAG	UGC	GAC	GAG	UUC	GGC	CAC	AUC	AAG	UUG		AAC	CCC	CAG	CGC	UCC	ACC	GUC		UAC
2	GCA	UGA					GGA		AUA		CUU			CCA		CGA	UCA	ACA	GUA		
3	GCG						GGG				CUC			CCG		CGG	UCG	ACG	GUG		
4											CUA					AGA	AGU				
5											CUG					AGG	AGC				
Number of Bits	2	2	1	1	1	1	2	1	2	1	3	0	1	2	1	3	3	2	2	0	1

3.2 The Natural Encoding Algorithm

In our proposed Algorithm, each 6 bits from data in binary form is to be converted to the form of DNA using Table 2 then to amino acids' form. Each DNA triple is to be treated as a separate codon. Table 3 is used to represent its amino acid form by describing the codon's location in the table. The codon's location is to be defined by its amino acid (column) and its ambiguity (row).

The value of the amino acid (column) is represented by the character abbreviation of the selected amino acid. The ambiguity number is to be represented in the form of bits using different number of bits. The ambiguity bits will be added in front of the binary form of the rest of the data. The next step is to convert the next corresponding 6 bits into DNA form and then to amino acids' form as explained, and so on till reaching the end of the data. By this, the ambiguity of one step is added to the data to be encoded in the next step. The algorithm is illustrated in the following pseudo code Figure 2.

It cannot be estimated initially how many bits will be processed or the number of bits will be left at the end because they depend on the nature of each amino acid involved in the data. The algorithm processing should be sequential as each step is dependent on the previous step. Each 6 bits produce one amino acid in addition to its ambiguity bits. The next step also collects 6 bits as input (the resulted ambiguity of the previous step (1, 2 or 3 bits) + input data (5, 4, or 3 bits) respectively).

```
Input: binary data (Inp)
Output: amino acid form of data (Out), remainder bits (Rem)
Processing:
    1- while (Inp.length >= 6)
        a. (B)= take first 6 bits from Inp.
        b. (D)= convert B to DNA form using
           table 1 (result is 3 DNA)
        c. Search table 3 to find the cell
           containing D, and select:
              a. (A): amino acid column
              b. (Ambig): bits defining the
                 level of ambiguity
        d. Add A to Out
        e. Add Ambig to Inp (at the beginning)
    2- If (Inp.length < 6)
Then (Rem)= Inp
```

Fig 2: Algorithm of the Proposed Natural Encoding to Amino Acids' form

But on reaching the last bits which are less than 6 bits, we call these bits, the remainder. The reminder bits –whose number varies between 0, 1, 2, 3, 4, 5 bits - are too little to be processed as their size is less than the size of one amino acid. Consider the following example to encode the word "FCIS" whose Hexadecimal representation is "46 43 49 53" and bits representation is "01 00 01 10 01 00 00 11 01 00 10 01 01 01 00 11".

The input in the form of 32 bits is converted into 7 amino acid characters "HBHSPGC" leaving a remainder 2 bits "11". The encoding details are explained in Table 4.

Table 4: The steps of Natural Encoding of the word "FCIS" to the form of Amino Acids.

Iteration	Bits	DNA	Amino acid	Ambiguity
1	10001	CAC	H	1
2	**1** 10010	UAG	B	01
3	**01** 0001	CAC	H	1
4	**1** 10100	UCA	S	010
5	**010** 100	CCA	P	10
6	**10** 1010	GGG	G	11
7	**11** 1001	UGC	C	1
8	**1 1**			

In order to define the Input to Output ratio, we have to consider the best and worst cases as the number of ambiguity bits varies according to the nature of the amino acids of the data. If the input size is (N) bits, the best case is when ambiguity size is always the minimum or equals to zero. In this case, the output size = 8/6 N = (4/3 N). This is because each 6 bits produce one amino acid character which corresponds to 8 bits in output so Input: Output ratio = 3:4. In this case the output has the minimum size.

The worst case is when the ambiguity size is the maximum or equals to 3 bits. If the input size is N, the output will be the result of processing (N+1/3N) bits. In this case, the output size will be $\frac{4}{3} \times \frac{4}{3} N = \frac{16}{9} N$. This is because if input size is N, it is increased by 1/3N representing the ambiguity to be N+ 1/3N = 4/3N. So the output size is the size of the processed 4/3 N. the input to output ratio will be 9:16 and this is the case of the maximum size that can reached by the output.

Best case	**Input / Output = ¾**
Worst case	**Input / Output = 9/16**

3.3 The Decoding Algorithm (Reverse Encoding)

In order to decode the amino acid form of data to get the initial binary form, we have to start from the end of the amino acid sequence in addition to the remainder. The last amino acid is selected to know the number of bits used to define its ambiguity. These bits are extracted from the remainder. If there are any other bits in the remainder, then they are put in the end of the binary output. The amino acid along with the ambiguity bits are used to substitute in Table 3, and get the desired codon (DNA triple). The 3 DNA's are converted to binary form using Table 1. Then the result is put in the remainder and so on till the data is finished. The algorithm details are explained in Figure 3.

```
Input: Amino acid form of data (AA), Remainder bits (R).
Output: Binary Data (B)
Processing:
(Len) = AA.length
While Len >0
    1- (SAA) = AA[Len]    //selected amino
       acid which is the last one in the
       stream
    2- (NB) = get_num_bits(SSA) // search
       table 3 to get the number of bits for
       SAA
    3- (Ambig) = extract the first NB bits
       from R
    4- R= remove the first NB bits from R
    5- Use SAA and Ambig to substitute in
       table 3 and get the corresponding
       codon (D)
    6- B=R+B
    7- (SB) = convert D into bits
    8- R= SB
    9- Len --
End while.
```

Fig 3: Algorithm of Amino Acids' Natural Decoding

The following table 5 shows in details how to decode the amino acid data resulted from the previous example back to the initial binary form. The amino acid data is "HBHSPGC" and remainder is "11". We will start from the last amino acid till the first one.

The result is "0100 0110 0100 0011 0100 1001 0101 0011" which is the same input to the encoding algorithm explained in the previous example. It is the same Hexadecimal form of the word "FCIS". The algorithm reversibility is proved through implementation. As we can see, each iteration in the decoding algorithm depends on the output resulted from the previous iteration. That is why the processing of the algorithm should be sequentially implemented and cannot be implemented in parallel).

Table 5: The steps of natural amino acid's decoding of the example of in Table 4

Iteration	Amino acid	Remainder	Ambiguity	Codon	Bits now
1	C	11	1	UGC	1
2	G	111001	11	GGG	1001 1
3	P	101010	10	CCA	1010 1001 1
4	S	010100	010	UCA	100 1010 1001 1
5	H	110100	1	CAC	10100 100 1010 1001 1
6	B	010001	01	UAG	000110100 100 1010 1001 1
7	H	110010	1	CAC	10010 0001 10100 100 1010 1001 1
8	-	010001	-	-	010001 10010 0001 10100 100 1010 1001 1

We succeeded in implementation of an encoding algorithm that translates any binary data into the environment of 21 amino acids, without losing any of the initial data and with zero errors. This was proved by the algorithm's reversibility.

3.4 Algorithm Complexity

The encoding algorithm of input size (N) runs in a loop that processes 6 bits per iteration. But this input size increases during each iteration with constant C= 0, 1, 2, or 3 bits. Assume that m is the number of iterations till the algorithm halts. At each iteration, i which is initially zero is incremented by 6 in each iteration, so in total, it will be increased by 6m. N is increased by C in each iteration, so N will be increased by Cm. The halting condition is when i == N which is equivalent to 6m = N + Cm for m iterations. Solving this equation, m = N/ (6-C). Then the total processing time is **T (N/ (6-C)) for C < 6 or the complexity is O (n)**.

3.5 Implementation

We succeeded in implementing the algorithm on a computer program using C#. For example, we will consider the input "**32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34**" in the hexadecimal format. It is converted to binary which resulted in 128 bits then converted to DNA/RNA form using Table 1. The result is "**AUA GCA AUU UCG GGG AGA GAC CGG AUA AGA UCA UAC AUA CGC GAG GAG UGA AAU CUA ACU AUC A**". Afterwards, Conversion to Amino acids form is implemented using our proposed algorithm. The resulted amino acids= "**I G Q L P R L R P A E W I E A C Y A Y T B R P L E T L R D H**" with remainder "00100". Figure 4 shows screenshot of the program implementation.

4. DISCUSSION AND ANALYSIS

The introduced algorithm proved to meet the main characteristics of an algorithm. Definiteness: the algorithm is clearly specified and implemented through a computer program. Effectiveness: steps are sufficiently simple, basic and easily reversible. Input is defined to be any sequence of binary data. Output is defined to be a sequence of English characters representing the Amino acids in additional to remainder in the form of bits. All the outputs can be represented in a binary form. Finiteness: the algorithm terminates after a finite number of steps which is proved in the algorithm complexity. We have proved that the encoding algorithm is reversible and applicable.

Moreover, the natural algorithm can be implemented with one or many rounds. The idea of representing the amino acid in English characters makes this form to be used as input to additional rounds. This is implemented by calculation the ASCII code of each letter. Then we can convert it to the binary then DNA forms which act as input to a new round.

In the field of cryptography, the encoding techniques cannot provide security by their own as they don't include the use of a secret key [11]. But they can be embedded into another encryption algorithm to enhance confusion specially that the output can be again represented in digital form which is completely different from the input. This was successfully implemented in a previous paper as a hybrid system with a cryptographic algorithm [9].

The sequence of characters in the input message clearly affects the output. This is because it is based on combining triples of DNA while one character is represented by 4 DNA. Also because of the inclusion of ambiguity bits whose number varies between 0, 1,2 or 3 bits according to the nature of the amino acid. These all allow interfering of code between successive characters.

This concept is suitable for applying data integrity, digital signature and confidentiality as the change of a portion of the message will lead to variance in the output which propagates to the rest of the output. The remainder is a very critical member in the process of decoding that it is considered the key to decode the message. The loss of the remainder for instance will make us completely unable to decode the message.

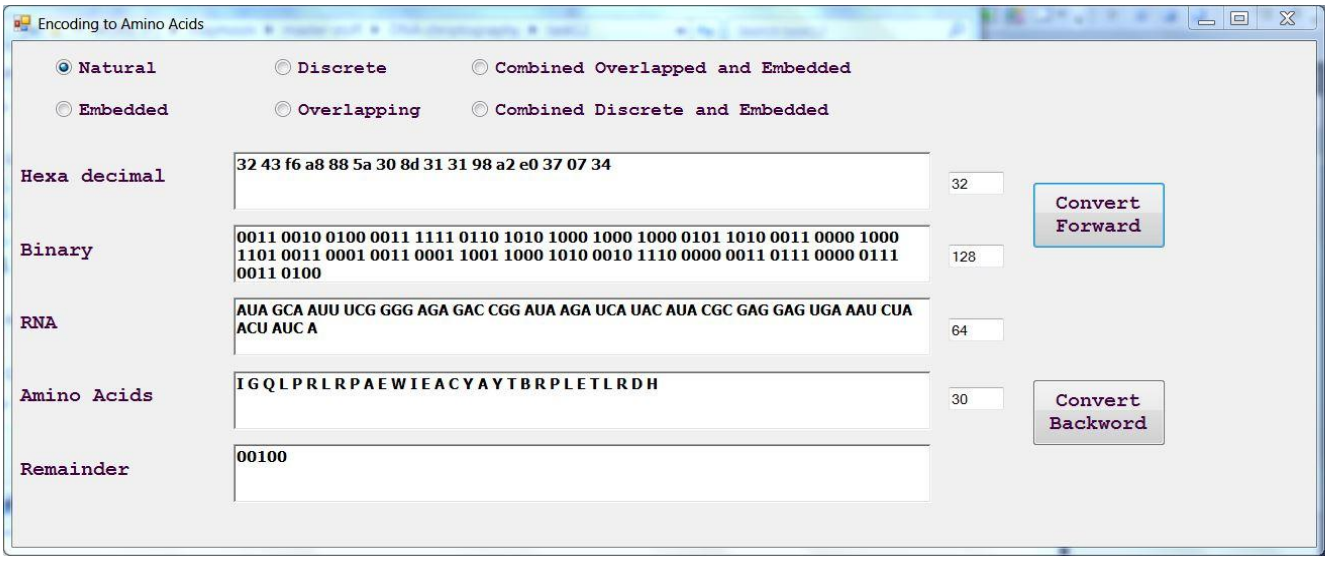
Although the algorithm uses the natural distribution of codons on amino acids, but we cannot say that the resulted form is a direct and natural representation of amino acid form of data. This is because of including the ambiguity inside the encoding process. The result is that not each bit presented in the amino acid, represents a bit in the initial data.

The algorithm translates any digital data into the environment of 21 amino acids, without losing any of the initial data and with zero errors. This was proved by the algorithm's reversibility.

5. CONCLUSION

Our proposed encoding method used the natural distribution of codons on the 20 natural amino acids to develop a reversible method of data encoding from binary form to DNA then to the form of Amino acids. The encoding algorithm and its decoding proved to meet the main characteristics of an applicable reversible algorithm. They can be implemented with one or many rounds. This concept is suitable for applying data integrity, digital signature and confidentiality.

The importance of such encoding algorithm lies mainly in representing data in a biological form that can make data be able to go through biological experiments and processes, especially related to Amino Acids and DNA. It is also a way of viewing data moving through biological processes and representing it in a binary form which can be used in many computer applications. As they don't include a secret key they cannot provide security by their own. But they can be combined with other traditional or biological cryptographic algorithm to create new security systems.



[✓ Natural] Discrete Combined Overlapped and Embedded
Embedded Overlapping Combined Discrete and Embedded

```
Hexa decimal   32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34                          32

Binary         0011 0010 0100 0011 1111 0110 1010 1000 1000 1000 0101 1010 0011 0000 1000
               1101 0011 0001 0011 0001 1001 1000 1010 0010 1110 0000 0011 0111 0000 0111
               0011 0100                                                                 128

RNA            AUA GCA AUU UCG GGG AGA GAC CGG AUA AGA UCA UAC AUA CGC GAG GAG UGA AAU CUA
               ACU AUC A                                                                 64

Amino Acids    I G Q L P R L R P A E W I E A C Y A Y T B R P L E T L R D H              30

Remainder      00100
```

Fig 4: Program Implementation of Natural Encoding.

6. REFERENCES

[1] Sabry Mona, Hashem M., Nazmy Taymoor, September 2012, Three Reversible Data Encoding Algorithms based on DNA and Amino Acids Structure. International Journal of Computer Applications (IJCA), 54(8):24-30, Published by Foundation of Computer Science, New York, USA.

[2] Kari L., 1997, DNA Computing: Arrival of Biological Mathematics," The Mathematical Tntelligencer, 19, pp. 9–22.

[3] Kartalopoulos S.V., 2005, DNA-inspired cryptographic method in optical communications, in Authentication and Data Mimicking Military Communications Conference 2005, pp. 774–779.

[4] Lu M. X., 2007, Symmetric-key cryptosystem with DNA technology, Science in China Series F: Information Sciences, vol. 3, pp. 324–333.

[5] Gehani A., LaBean T. H. and Reif J. H., 2000, DNA-based cryptography, DNA Based Computers V. Providence: American Mathematical society, vol. 54, pp. 233–249.

[6] Celland C. T., Risca V. and Bancroft C., 1999, Hiding messages in DNA microdots, Nature, vol. 399, pp. 533–534.

[7] KANG Ning, October, 2004, A Pseudo DNA Cryptography Method, Independent Research Study Project for CS5231.

[8] Heider Dominik and Barnekow Angelika. 2007, DNA-based watermarks using the DNA-crypt algorithm. BMC bioinformatics, 8(1):176.

[9] Sabry Mona, Hashem M., Nazmy Taymoor, Khalifa M.E., 2010, A DNA and Amino Acids-Based Implementation of Playfair Cipher, International Journal of Computer Science and Information Security (IJCSIS), 8(3).

[10] Nixon, D., 2002, DNA and DNA Computing in Security Practices – Is the Future in Our Genes? GSEC Assignment Version 1.3, SANS Institute.

[11] Stallings W., 2003, Cryptography and Network Security, Third Edition, Prentice Hall International.