A DNA-based Implementation of YAEA Encryption Algorithm

Sherif T. Amin [1], Magdy Saeb [2], Salah El-Gindi[1]
1. Dept. of Mathematics, Faculty of Science, Assiyut University,
2. Arab Academy for Science, Technology and Maritime Transport,
School of Engineering, Computer Dept., Alexandria, Egypt
E-mail: mail@magdysaeb.net

ABSTRACT
The fundamental idea behind this encryption technique is the exploitation of DNA cryptographic strength, such as its storing capabilities and parallelism in order to enforce other conventional cryptographic algorithms. In this study, a binary form of data, such as plaintext messages, and images are transformed into sequences of DNA nucleotides. Subsequently, efficient searching algorithms are used to locate the multiple positions of a sequence of four DNA nucleotides. These four DNA nucleotides represent the binary octet of a single plaintext character or the single pixel of an image within, say, a Canis Familiaris genomic chromosome.
The process of recording the locations of a sequence of four DNA nucleotides representing a single plain-text character, then returning a single randomly chosen position, will enable us to assemble a file of random pointers of the locations of the four DNA nucleotides in the searched Canis Families genome.
We call the file containing the randomly selected position in the searchable DNA strand for each plain text character, the ciphered text.
Since there is negligible correlation between the pointers file obtained from the selected genome, with its inherently massive storing capabilities, and the plain-text characters, the method, we believe, is robust against any type of cipher attacks.

KEY WORDS
DNA, encryption, decryption, cryptography, algorithm, data communication security

1. Introduction

The investigation conducted in this paper is based on a conventional symmetric encryption algorithm called "Yet Another Encryption Algorithm" (YAEA) developed by Saeb and Baith [1]. In this study, we introduce the concept of using DNA computing in the fields of cryptography in order to enhance the security of cryptographic algorithms. It was Adleman, with his pioneering work [Adleman, 1994]; who set the foundation for the new field of bio-computing research. His main notion was to use actual chemistry to solve problems that are either unsolvable by conventional computers, or require a massive amount of computation. Gehani et al., [2] introduced the first trial of DNA-based cryptography.
In this research work, we are not determined to use real biological DNA strands for computing, but just the principle ideas of the central dogma of molecular biology. The method simulates the transcription, splicing, and translation process of the central dogma; thus, it is a virtual DNA cryptographic method. This work is organized as follows:
Section II presents the design rationale, a formal description of the algorithm is represented in section III, section IV demonstrates the hardware applicability of the algorithm, and finally section V provides a discussion of the results. In addition, we provide a summary and our conclusions.

2. Design Rationale

DNA has the potential to be a long-term dense information storage system, due to the following reasons:

- Massive amounts of data can practically be stored in a high-fidelity compound on the molecular level. A single gram of dried DNA is capable of storing the same amount of information that could fit on one trillion CDs [7].
- Despite changing environmental and technological conditions, DNA has lasted as a storage molecule for millions of years.
- DNA stores our own human genetic information and thus will be critical to maintain for the near future.
- Replication of identical copies of the information storage medium is relatively easy if DNA is used [7].

DNA computing guarantees massive parallelism, with a proper setup and enough DNA, huge problems can potentially be solved by parallel search. This can be much faster than a conventional computer, for which massive parallelism would require large amount of hardware, not simply more DNA.
A Canis Familiaris genome was downloaded from GenBank. In our case Canis Familiaris chromosome: (gi|73948581|ref|NW_876270.1|Cfa1_WGA2_2 Canis

familiaris chromosome 1 genomic contig) is delivered from the sender to the receiver of the encrypted message through a secured channel. The binary form of information of the plain-text message at the sender end, is transformed into DNA form (A for 00, C for 01, G for 10, T for 11) resulting in a single DNA strand that represents the message. Explicitly, each four DNA nucleotides sequence represents a binary octet that in turn represents a plain-text character of the message. The sender will match all the quadruple DNA nucleotides sequence representing all the plain-text characters against the single strand DNA representing the Canis Familiaris genome in order to locate all of its occurrences and positions or in other words its pointers. For example, a sequence such as CAGA, which represents the capital plain-text character H, appears 391919 times in our Canis familiaris DNA strand of length 53004996 base pair (bp). This would give us 391919 possible locations for the same. Subsequently, the sender randomly retrieves from the single Canis DNA strand one location coordinate for each quadruple DNA nucleotides sequence representing a plain-text character for all characters in the message. This process will generate a file that contains random location pointers for all plain-text characters. We call this file the ciphered text. The ciphered text is then transmitted to the receiver. Upon receiving this file, the receiver uses the ciphered text to recover the sequences of quadruple DNA nucleotides sequence from the Canis DNA strand, which was previously delivered to him through a secured channel. In the reverse order, the recovered DNA form of information will be used to retrieve the binary form of information, which will be then translated into plain text.

3. The Algorithm

The algorithm utilizes a sequential search algorithm in order to locate and randomly return one of the many positions of quadruple DNA nucleotides sequence representing the binary octets of plain-text characters. A one round of the algorithm is formally summarized as follows:

ALGORITHM YAEADNA

[The algorithm uses a search technique in order to locate and return the position of quadruple DNA nucleotides sequence representing the binary octets of plain-text characters.]

Input: *A [plaintext character], M [Random binary file], RND [G], .DNA* nucleotides sequence*,*

Algorithm Body:

1. In a plain text file, each character is sequentially replaced by its ASCII code:

A → ASCII [A];

2. Replace ASCII code by its DNA sequence:

ASCII[A] → DNA sequence;

3. Starting from a random location in a binary file represented in the form of a single strand DNA sequence, one searches for a quadruple DNA nucleotides sequence representing a plain text character.

This "sought-after-sequence" has the same nucleotides sequence as the ASCII code of the plain-text character.

4. The sequential search is performed starting from a random location X.

X → RND [G];

5. If the correct pattern is found, its location is then recorded in a pointer (PTR) or locations' output file.

Start sequential search for sequence SEQ from location X;

This output file is called, rather inaccurately, the ciphered text. However, the file contains only pointers to the location of the found octets.

If DNA [A] = DNA [SEQ] then generate PTR;

6. Repeat the same procedure, starting from step **2,** for all other characters;

Output: PTR, a pointer to the location of the found quadruple DNA nucleotides sequence representing the binary octet.

End Algorithm.

The decryption process is achieved by using the pointer file and the same random binary file that is available to both send and receiver in advance. A summary of this method is shown in Figure 1. For even higher degrees of security, the algorithm is repeated for an "in advance agreed-upon" number of rounds. However, this second round is hardly required for most ordinary-level security applications. The huge storage capability, even for relatively small strand of DNA as will be shown in section IV, provides the required echelon of immunity against frequency attacks. The searched patterns can be easily found in a range from tens of thousands of times to hundreds of thousands of times. This point will be methodically investigated in the section on algorithm security analysis. We will show that there is negligible, if any, correlation existing between the pattern locations (the cipher) and the plain text characters. Moreover, we apply the proposed algorithm on images to try to investigate if there are any visible patterns in the ciphered image.

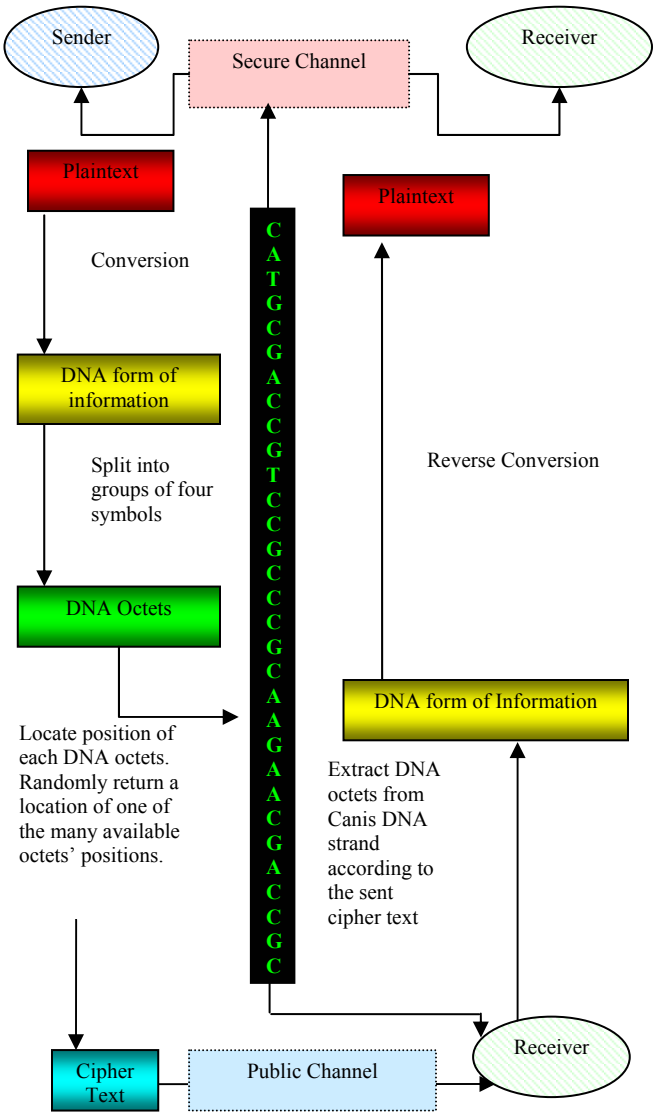

Figure 1: The Computational graph of one YAEADNA round.

4. Security Analysis of the Algorithm

The entire novel "Uncle Tom's Cabin" was downloaded from project Gutenberg [http://www.gutenberg.org/] and used for encryption. Table 1, shown below, illustrates all 82 characters occurring in the novel, and their total frequency in the Canis familiaris chromosome 1.

As shown in the table 1, Canis Familiaris with its 53,004,966 bp DNA sequence in chromosome 1 has the ability to encode all the novel's plaintext characters within its strand sequence due to the abundance of nucleotides combination. However, for higher levels of security one can always resort to larger genome sequence.

Char.	Freq. in DNA	Char.	Freq. in DNA
[l]	23618	[E]	210222
[f]	25392	[/]	210841
[a]	27895	[.]	216379
[6]	34295	[B]	216498
[c]	34984	[G]	217607
[o]	35244	[K]	217891
[m]	39016	[:]	223088
[k]	40134	[S]	223518
[d]	41598	[I]	224830
[h]	43075	[%]	225581
[[]	43545	[4]	225977
[X]	46915	[)]	229001
[&]	47593	[Q]	229883
[g]	47631	[P]	231686
[v]	50643	[7]	247484
[b]	50722	[N]	249825
[i]	51632	[;]	253181
[Y]	60091	[?]	253181
[e]	61618	[3]	259677
[n]	62262	[_]	261141
[F]	62918	[8]	264718
[j]	91953	[]]	265877
[V]	92378	[y]	267487
[q]	118728	[O]	274382
[r]	130831	[x]	277367
[A]	150386	[D]	281713
[1]	150665	[t]	282656
[-]	164485	[u]	287690
[,]	164592	[W]	291964
[s]	166484	[(]	306117
[2]	167323	[z]	313004
[p]	171488	[R]	313137
[9]	171570	[T]	313743
[>]	185988	[0]	313988
[C]	187115	[@]	332166
[L]	190084	[w]	336562
[5]	194813	["]	338352
[M]	198309	[^]	347070
[!]	199178	[J]	349870
[U]	202537	[ ]	391919
[']	208559	[H]	391919

Table1: Plain Text Characters Frequency in DNA Strand

5. Verification of the algorithm security

In order to detect if there are any indirect relationships between locations of all the plain text characters in the selected text, the novel, and their randomly selected locations in the DNA strand, a correlation analysis was performed for all the 1,015,120 novels' characters . The results of Pearson correlation analysis, shown in Figure 2, indicate that the majority of the locations of the novel's alphabetical characters lack any significant relationship with their DNA locations. Seventy-one characters out of 82 (total number of characters occurring in the novel) have their Pearson correlation coefficient residing between -0.1 and 0.1. The other eleven characters, showed correlation factor between -0.12 and 0.48.

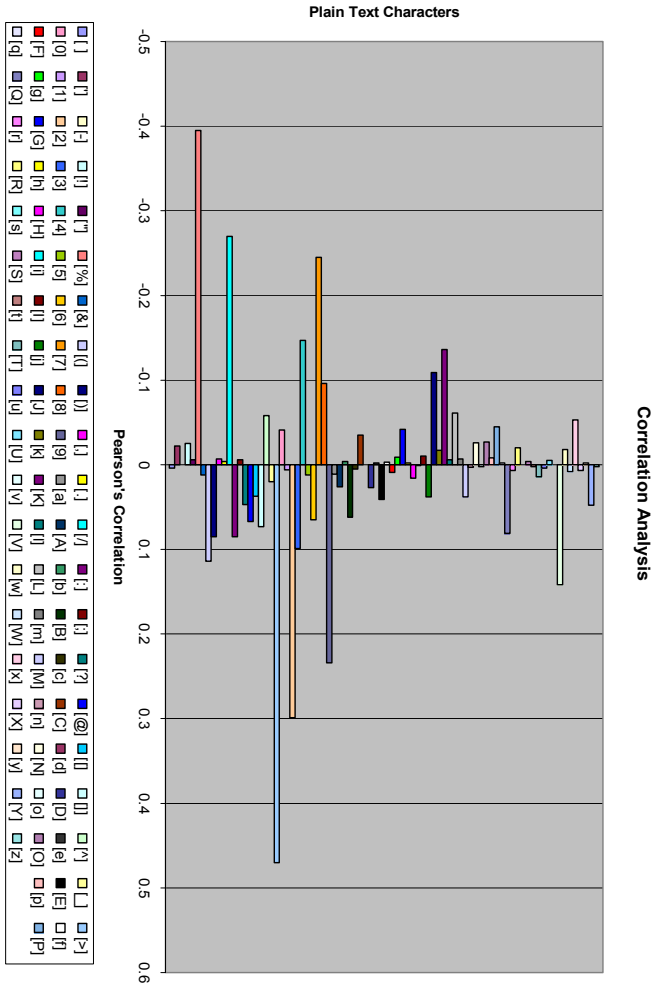

Figure 2: Pearson's Correlation Analysis between the cipher and plain texts

6. Application on images

We tested the encryption process on images to show how random is the selection of DNA octet's locations on the encrypting sequence. The picture below (trees.TIF) (350x258) was used as the test image. After transforming trees.TIF into a single DNA strand, searching and retrieving the locations of the trees DNA octets into an 88,410,189 nucleotides dog chromosome, the 90300=350x258 retrieved locations elements were reshaped as a 350x258 matrix, then their value rescaled to the full range of the current color map. The encrypted image is shown below.

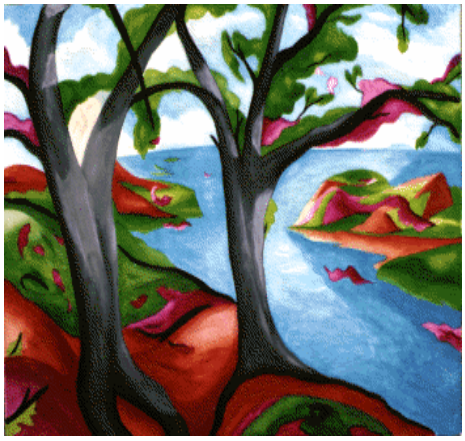

Figure3: Test image (trees.TIF)

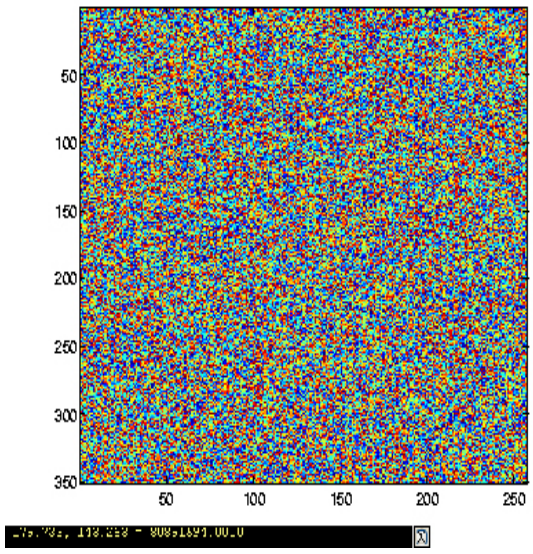

Figure4: The encrypted image

7. Summary and Conclusion

In this article, we have described a symmetric DNA-based cipher approach. This method is effortlessly scalable for large digital information products. We have shown that the algorithm is effective at encrypting and decrypting digital information from biological DNA strand.

In addition, the proposed methodology could easily be improved using a larger DNA strand that already exists. The special features of the proposed cipher are summarized as follows:

- The algorithm is a symmetric cipher consisting of recording pointers to the randomly selected locations of the file in the searchable DNA strand for each plain text character.
- We have demonstrated that, even in a relatively simple DNA strand such as found in the Canine Families, the patterns found can range from tens of thousands of times to hundred of thousands. Consequently, reducing or probably eliminating altogether the vulnerability to frequency attacks. This point was clearly demonstrated using Pearson correlation coefficient.
- A test was conducted by recording the time needed to encrypt the "Uncle Tom's Cabin" novel using six different DNA strands of different lengths. Utilizing a dedicated 3.2-GHz CPU employing 3G RAM, we ran this test 1000 times for each individual DNA strand, then took the average time value. The table shown in appendix illustrates the results obtained in seconds. The results shown in this table and the accompanying graph clearly illustrate that the algorithm has high throughput capability that ranges between 0.3 to 1.2 microseconds per nucleotide.

Based on the above discussion, one can conclude that the security and the performance of the proposed algorithm are satisfactory for multi-level security applications of today's networks.

References

[1] M. Saeb, A. Baith, "An Encryption Algorithm for Data Security," Recent Advances in Information Science & Technology, N.E. Mastorakis, (editor), *World Scientific Publishing Company*, pp. 350-354, 1998.

[2] Ashish Gehani, Thomas LaBean and John Reif. "DNA-Based Cryptography. DIMACS DNA Based Computers," V, *American Mathematical Society*, 2000.

[3] A. Leier, C.Richter, W. Banzahf, H. Rauche, " Cryptography with DNA Binary Strands," University of Dortmund, *School of Computer Science*, December, 1999.

[4] Leonard Adleman, "Molecular Computation of Solutions to Combinatorial Problems," *Science*, 266:1021-1024, November 1994.

[5] Aluízio Borém, Fabrício R. Santos, David E. Bowen *Understanding Biotechnology* Prentice Hall, 1998.

[6] Bruce Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition* John Wiley & Sons, 1996.

[7] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, Peter Walter, *Molecular Biology of the Cell, Fourth Edition* Garland Publishing, 2002.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, *Introduction to Algorithms, Second Edition* The MIT press, 2001.

Appendix

Length of DNA strand in nucleotides	Average time of 1000 runs in seconds	Average time per nucleotide in micro seconds
19,073,834	23.1543136	1.21393
23,771,897	12.0133299	0.505358
26,542,582	21.9985904	0.828804
30,915,115	23.0500547	0.745592
53,004,996	21.9193091	0.413533
88,410,189	28.2153203	0.319141

Table A.1 Average run times for Various strand lengths