I, Anusha Mantha, hereby submit this original work as part of the requirements for the degree of Master of Science in Computer Engineering.

It is entitled: Improving Reliability in DNA based Computations with Applications to Cryptography

Student’s name: Anusha Mantha

This work and its defense approved by:

Committee chair: Carla Purdy, PhD
Committee member: Raj Bhatnagar, PhD
Committee member: Wen Ben Jone, PhD
Improving Reliability in DNA based Computations

with Applications to Cryptography

A thesis submitted to the

Graduate school of the University of Cincinnati

in partial fulfillments of the requirements for the degree of

**Master of Science**

in the School of Electronics and Computing Systems

of the College of Engineering and Applied Science

October 2012

By

**Anusha Mantha**

MSc. in Chemistry, Birla Institute of Technology & Science, Pilani, India 2008

B.E. in Electrical & Electronics Engineering, Birla Institute of Technology & Science,

Pilani, India 2008

**Thesis Advisor and Committee Chair: Dr. Carla Purdy**
Abstract

DNA computing has become an attractive alternative to traditional computing techniques due to the large storage capacity of DNA, massive parallelism and energy efficiency of the computations. However, since DNA computing is based on biomolecular reactions, it is not error free. One of the main sources of these errors is false hybridizations between unintended sequences during the computation process. An efficient way to minimize these errors is through designing error-resistant DNA sequences. The DNA encoding problem is the design of DNA sequences in such a way that every DNA sequence hybridizes with its own complementary sequence during the computation process. In this thesis, the DNA encoding problem has been studied and approached using the simulated annealing algorithm. Several combinatorial constraints have been considered to optimize the DNA sequences so that they are resistant to false hybridizations during the computations. A cost function which captures the DNA design requirements has been developed and minimized using simulated annealing algorithm. Also, application of DNA computations to the field of cryptography has been studied. A new cryptographic algorithm based on linear self-assembly of DNA has been proposed in which the rate of hybridization errors could be controlled using the error-resistant DNA sequences generated using the simulated annealing algorithm in this work. A simulation framework to simulate the DNA cryptography algorithms has been developed using an object oriented approach. The DNA self assembly based cryptography algorithm developed in this work and an existing DNA cryptography algorithm based on polymerase chain reaction have been simulated and compared in terms of the implementation time complexity.
To My Family
Acknowledgements

To start with, I would like to thank God for all that he has given me and made possible in my life.

For my work to reach a point where it stands today it needed a great deal of knowledgeable mentorship, guidance and patience and I could not have asked for a better advisor. I am deeply indebted to Dr. Carla Purdy to have accepted me in her research group. My association with her throughout my work has certainly improved my process of thought and application of ideas and I always left her office with a sense of satisfaction after our productive discussions.

I express my sincere gratitude to Dr. Raj Bhatnagar and Dr. Wen-Ben Jone to have consented to be in my defense committee. I would like to extend my thanks to them for taking out time from their busy schedules.

I always believed in myself, but my husband, Vivek, believed in me more. His constant support and occasional pep talks helped me often give my best and go the extra mile. My whole family needs a special mention. They never ceased to show the love and affection that kept me going through situations, both good and bad.

If my academics made my stay in Cincinnati purposeful, it was my friends that made it enjoyable and memorable. I would like to thank Bharat, Radhika, Venkat and Vignesh for being there for me in times of need in the last 2 years.
# Table of Contents

1. **Introduction** .................................................................................................................. 1
   1.1 Overview ............................................................................................................................ 1
   1.2 Goals of the thesis ............................................................................................................ 2
   1.3 Thesis outline ................................................................................................................... 2

2. **Background** ..................................................................................................................... 4
   2.1 The molecular structure of DNA ....................................................................................... 4
      2.1.1 Structure of nucleotide ............................................................................................... 4
      2.1.2 Formation of single stranded DNA ............................................................................. 5
      2.1.3 Watson-Crick complementarity .................................................................................. 6
      2.1.4 Formation of double stranded DNA .......................................................................... 7
      2.1.5 Length of DNA molecule ........................................................................................... 8
   2.2 DNA operations ................................................................................................................ 8
      2.2.1 Merge operation .......................................................................................................... 8
      2.2.2 Denaturation and annealing ...................................................................................... 9
      2.2.3 Cutting ....................................................................................................................... 9
      2.2.4 Ligation ...................................................................................................................... 10
      2.2.5 Separation based on length ...................................................................................... 11
      2.2.6 Extraction .................................................................................................................. 12
      2.2.7 Polymerase Chain Reaction (PCR) ............................................................................ 13
2.3 DNA computation models .................................................................................. 14
  2.3.1 Adleman’s unrestricted model .................................................................. 15
2.4 Errors in DNA computation ............................................................................ 19
  2.4.1 Annealing ................................................................................................. 19
  2.4.2 Ligation .................................................................................................... 20
  2.4.3 PCR ......................................................................................................... 20

3. DNA cryptography .............................................................................................. 21
  3.1 Overview ...................................................................................................... 21
  3.2 Comparison of DNA cryptography to traditional cryptography ............... 24
  3.3 Previous work in DNA cryptography .......................................................... 26
    3.3.1 Cryptography based on PCR of DNA .................................................... 27

4. DNA self-assembly based cryptographic algorithm ............................................ 30
  4.1 One-time pad generation .............................................................................. 31
    4.1.1 DNA coding of binary numbers ............................................................ 31
    4.1.2 Implementation using DNA operations ............................................... 32
  4.2 Encryption .................................................................................................... 33
  4.3 Decryption .................................................................................................... 34
  4.4 Implementation issues .................................................................................. 34
  4.5 Evaluation of the cryptographic algorithms ............................................... 35

5. Generating reliable DNA sequences ................................................................ 37
  5.1 Overview ...................................................................................................... 37
5.2 Hybridization errors ........................................................................................................... 38
5.3 DNA encoding problem ...................................................................................................... 41
5.4 DNA sequence design considerations ............................................................................. 42
  5.4.1 Hamming distance constraint ..................................................................................... 42
  5.4.2 H-Measure .................................................................................................................. 43
  5.4.3 Cost function ............................................................................................................... 44
  5.4.3 Combinatorial constraints for the DNA encoding problem ...................................... 46
5.5 Simulated Annealing (SA) approach ............................................................................... 46
  5.5.1 Description .................................................................................................................. 46
  5.5.2 Implementation .......................................................................................................... 50
  5.5.3 Results ....................................................................................................................... 50
5.6 Comparison with previous work ..................................................................................... 56
  5.6.1 Genetic Algorithm (GA) approach ............................................................................ 56
  5.6.2 Branch and Cut Algorithm (BCA) approach .............................................................. 57
  5.6.3 Evolutionary Algorithm (EA) approach .................................................................... 58
  5.6.4 Comparison results ..................................................................................................... 59

6. Simulation of DNA cryptography algorithms .................................................................... 63
  6.1 Overview .......................................................................................................................... 63
    6.1.1 Implementation details .............................................................................................. 63
  6.2 Classes used and their relationships .............................................................................. 64
    6.2.1 Description of the classes used ................................................................................ 64
6.2.2 Class diagram ........................................................................................................... 69

6.3 DNA operations ............................................................................................................. 71

6.3.1 Implementation details ............................................................................................. 71

6.3.2 Comparing time complexities .................................................................................... 72

6.4 DNA cryptography algorithms ..................................................................................... 73

6.4.1 PCR based cryptography ........................................................................................... 73

6.4.2 XOR cryptography based on linear self assembly of DNA ..................................... 75

6.4.3 Results ....................................................................................................................... 79

7. Conclusions and future work ......................................................................................... 81

7.1 Conclusions ................................................................................................................ 81

7.2 Future work ................................................................................................................ 82

References ....................................................................................................................... 83
List of Figures

Figure 2.1: Chemical structure of the nucleotide and different bases [2] .................... 5
Figure 2.2: Symbolic representation of DNA single strand [5]................................. 5
Figure 2.3: Single strand of DNA formed by phosphodiester bonds [4] .................... 6
Figure 2.4: Structure of double stranded DNA molecule [5] ................................. 7
Figure 2.5: Double helical structure of DNA [6] ................................................. 7
Figure 2.6: Linear representation of DNA [5] ....................................................... 8
Figure 2.7: Denaturation and annealing operations [9] ......................................... 9
Figure 2.8: Cutting operation on DNA [8] ............................................................. 10
Figure 2.9: Ligation on DNA [10] ....................................................................... 11
Figure 2.10: Separation of DNA based on their length [9] .................................... 11
Figure 2.11: Extraction operation on DNA [8] ....................................................... 12
Figure 2.12: PCR technique for DNA amplification [11] ..................................... 14
Figure 2.13: Directed graph in Adleman's experiment. vin =0, vout=6 [12] .......... 16
Figure 3.1: Encryption and decryption process [17] [18] ..................................... 23
Figure 3.2: PCR based cryptosystem [20] ............................................................ 29
Figure 5.1: Complete hybridization between two strands .................................. 39
Figure 5.2: Hybridization with mismatches ......................................................... 39
Figure 5.3: Translocation hybridization ............................................................. 40
Figure 5.4: Hairpin structure formation by self hybridization of the DNA sequence ... 40
Figure 5.5: Translocation hybridization between two sequences with high Hamming 43
Figure 5.6: Flow chart for the simulated annealing algorithm .......................... 49
Figure 5.7: Fitness plot of temperature (T) versus total cost .............................. 51
Figure 5.8: Plot of time (sec) versus number of DNA sequences 'n' for L = 20
Figure 5.9: Plot of time (sec) versus length of DNA sequence (L) for 15 DNA sequences
Figure 5.10: Comparison of SA with BCA [31] and GA [34]
Figure 5.11: Comparison of SA with EA [32]
Figure 6.1: Representation of DNABase class
Figure 6.2: Representation of ssDNA class
Figure 6.3: Representation of dsDNA class
Figure 6.4: Representation of ssDNAOperations class
Figure 6.5: Representation of dsDNAOperations class
Figure 6.6: Representation of generateDNAEncodings class
Figure 6.7: Representation of BinPCRCryptography class
Figure 6.8: Representation of xorCryptography class
Figure 6.9: Class diagram showing the relationship between different classes
Figure 6.10: Sample output for PCR based encryption
Figure 6.11: Sample output for PCR based decryption
Figure 6.12: Screenshot of sample codebook of DNA sequences generated using the generateDNAsequences function
Figure 6.13: Screenshot of sample DNA sequences generated using the generateEncodingSeq function
Figure 6.14: Screen shot of sample output of the KeyGeneration function
Figure 6.15: Cipher text constructed from the DNA sequences after XOR encryption
Figure 6.16: Plain text constructed after the XOR decryption
List of Tables

Table 3.1: Comparison between modern and DNA cryptographic techniques [19]...... 25
Table 5.1: 14 reliable DNA sequences generated by SA approach.......................... 53
Table 5.2: Mean and standard deviation of data points from 10 runs for n=14........ 54
Table 5.3: 7 reliable DNA sequences generated by SA approach........................... 55
Table 5.4: Mean and standard deviation of data points from 10 runs for n=7........ 55
Table 5.5: DNA sequences from BCA [31] ............................................................... 59
Table 5.6: DNA sequences from GA [34] ............................................................... 60
Table 5.7: DNA sequences from EA [32] ............................................................... 60
Table 6.1: Time complexities in computer simulation, number of lab steps and real
time required for various DNA operations............................................................ 73
Table 6.2: Comparison results for PCR based and linear self assembly based
cryptography algorithms ....................................................................................... 80
Chapter 1

Introduction

This chapter introduces the work done in this thesis and outlines the thesis content.

1.1 Overview

With the advent of electronic computers, there came about a revolution in the field of computing. It is not long before its limitations in terms of miniaturization will be reached, at which point, one possible way to break this barrier would be to go to molecular levels. One of the main reasons to pursue molecular scale computing is to follow the minimization trend. Adleman’s experiment on solving Hamiltonian path problem using DNA set out the idea of using DNA for computational purposes [1].

Some of the characteristics of DNA that make it an attractive alternative for computational purposes are its ability to store information, high stability, massive parallelism during computations and energy efficiency. DNA computing has applications to several fields including cryptography. The field of DNA cryptography is in its developing stages, and new algorithms for data encryption are still being researched.

However, since biological operations are prone to several errors in spite of the well-defined procedures to carry out these operations, DNA computation is not error free. Hence, it becomes important to improve the reliability of DNA operations. The main source of error in DNA computation is false hybridization of unintended DNA sequences.
This thesis deals with minimizing these hybridization errors in DNA computations by generating reliable DNA sequences. The DNA sequence design problem is taken as a multi-objective optimization problem and several combinatorial constraints have been considered.

1.2 Goals of the thesis

The main goals of this research are

- To study different types of hybridization errors and generate reliable DNA encodings that minimize these hybridization errors
- To perform a survey on various cryptographic algorithms proposed so far and define metrics to evaluate the algorithms
- To develop a new cryptographic algorithm based on linear self assembly of DNA that is resistant to the hybridization errors during encryption and decryption
- To develop a framework for simulating the DNA cryptography algorithms using object oriented approach

1.3 Thesis outline

The organization of this thesis is given below.

Chapter 2 gives a background on DNA molecular structure, the DNA operations that are used in this thesis, Adleman’s experiment and a brief note on the sources of errors in DNA computation.

Chapter 3 discusses the application of DNA computing techniques to the field of cryptography. The DNA cryptography algorithms that were already implemented have been briefly explained.
In chapter 4, a new DNA cryptography method is presented and evaluated.

Chapter 5 deals with the possible hybridization errors in DNA computing and the problem of generating DNA sequences. A simulated annealing based approach has been proposed for generation of reliable DNA sequences for computations.

In chapter 6, an object oriented approach to DNA cryptographic algorithms is discussed.

In chapter 7, the conclusions and possible extensions to this work are discussed.
Chapter 2

Background

Since the DNA molecule is the central concept of DNA computing, it is important to study the molecular structure of the DNA molecule and how it can be manipulated to harness the computational capability of DNA [1].

2.1 The molecular structure of DNA

2.1.1 Structure of nucleotide

The single stranded DNA is a polymer chain made up of monomer units called nucleotides. Each nucleotide is comprised of a phosphate group and a nitrogenous base attached to a 5-carbon sugar named deoxyribose. Also, a hydroxyl group is attached to one of the carbons of the sugar structure which plays a part in deciding the orientation of the molecule. For convenience, the carbons on the sugar are labeled from 1’ to 5’. While the nitrogenous base is attached to 1’, the phosphate group is attached to 5’ and the hydroxyl (OH) group is attached to 3’.

There are four different bases which can attach to the sugar group and can be classified into purines and pyrimidines. Adenine (A) and Guanine (G) are the purines and Cytosine (C) and Thymine (T) are the pyrimidines. The nucleotides can thus be referred to as A, G, T, C depending on what base is attached to the sugar, as rest of the structure remains the same.
2.1.2 Formation of single stranded DNA

The nucleotides link together when the 5’-phosphate group of one nucleotide is joined with the 3’-hydroxyl group of the other forming a phosphodiester bond [1] and hence forming single stranded DNA molecules. This is depicted in Figure 2.3 [4]. Since the nucleotides are represented by their bases, the single stranded DNA can be represented as sequence of nucleotides A, G, T, C as shown in Figure 2.2 [5]. The phosphodiester bond imparts an inherent polarity to the DNA molecule [3]. The general convention to represent the orientation of a DNA strand is by writing the free 5’-phosphate group on the right and the free 3’-hydroxyl group on the left. This is referred to as 5’→3’ direction and can be seen in Figure 2.2 [5].

\[
\]

Figure 2.2: Symbolic representation of DNA single strand [5]
2.1.3 Watson-Crick complementarity

Watson-Crick complementarity is the cornerstone of understanding the structure and functioning of DNA [1]. The double-stranded DNA molecules are formed when the bases of two nucleotides interact with each other, forming weak, non-covalent bonds. However, the base pairing is restricted according to the Watson-Crick complementarity and can happen only between Adenine and Thyamine (A-T) and Guanine and Cytocine (G-C). Also, the base pairing happens in an anti-parallel orientation i.e., base at 5’-end of one strand is paired with base at the 3’-end of the other strand [3]. This can be observed in Figure 2.4 [5].
2.1.4 Formation of double stranded DNA

In reality, the DNA molecule is double helix in shape due to repulsion between the phosphate groups as shown in Figure 2.5 [6]. However, for the sake of simplicity, the double stranded DNA can be considered to be linear as shown in Figure 2.6 [5].
2.1.5 Length of DNA molecule

The length of a DNA single strand is determined by the number of nucleotides contained in it. A single strand of length ‘l’ is usually referred to as an ‘l-mer’. For a double stranded DNA, the length is determined by the number of base pairs (bp) present in the molecule.

2.2 DNA operations

In DNA computing, the DNA is utilized as a substrate for storing information and this information can be manipulated by performing several operations on the DNA [7]. The existing models of computations are based on various combinations of these DNA operations [8]. A detailed description of these operations can be found in [1] [8]. However, for the sake of the convenience of the reader, important operations used in this report are explained briefly in this section.

2.2.1 Merge operation

This is possibly the simplest of all operations where, given two test tubes, the contents of one test tube are poured into the other.
2.2.2 Denaturation and annealing

When the DNA solution is heated to a critical temperature, the double stranded DNA is separated into two single strands, since the weak bonds between the bases can be easily broken. This operation, known as denaturation, produces single strands of DNA while keeping their nucleotide sequence intact, which means that the single strands are still complementary to each other. Hence, when the heated solution is cooled down slowly enough, complementary base-pairing happens between the single strands to give rise to double stranded DNA. This operation is known as annealing. Both denaturation and annealing are extensively used in different models of DNA computing. Denaturation and annealing operations are shown in Figure 2.7 [9].

![Figure 2.7: Denaturation and annealing operations [9]](image)

2.2.3 Cutting

Restriction endonucleases are enzymes which cut the phosphodiester bonds at a specific recognition site and can be used to cut double stranded DNA molecules. These
enzymes recognize a particular short sequence of DNA, called the restriction site, and bind themselves at that site to cleave the DNA resulting in two DNA molecules with sticky ends. This operation is visually described in Figure 2.8[8].

**Figure 2.8: Cutting operation on DNA [8]**

2.2.4 Ligation

Ligation is the process of linking DNA molecules with compatible sticky ends which is mediated by the enzymes called ligases. This operation can be considered as the reverse of the cutting operation and is depicted in Figure 2.9 [10].
2.2.5 Separation based on length

The DNA molecules can be separated based on their length by gel electrophoresis. This technique involves applying an electric field to DNA molecules placed in a gel medium. In an aqueous solution, the negatively charged DNA molecules travel at the same rate, since the charge per unit length is constant. However, in the gel medium, the rate of migration is size dependent due to the fact that gel medium acts as a molecular sieve allowing the smaller molecules to pass easily compared to the longer molecules. This is depicted in Figure 2.10[9].

![Figure 2.10: Separation of DNA based on their length](image)
2.2.6 Extraction

Extraction is the process of separating out single strands of DNA containing a specific subsequence ‘α’ from a solution containing these as well as other strands. This can be achieved through a technique called affinity purification which involves attaching complementary sequences of ‘α’ (called probes) to tiny magnetic beads and passing the solution over these beads. The target molecules containing ‘α’ will get annealed to the probes and hence can be separated from the heterogeneous solution. This is shown in Figure 2.11 [8].

![Extraction operation on DNA](image)

**Figure 2.11: Extraction operation on DNA [8]**
2.2.7 Polymerase Chain Reaction (PCR)

Polymerase Chain Reaction (PCR) technique is used to solve the critical problem of amplification of specific DNA sequences in DNA computing. Also, PCR technique is very useful to detect known DNA fragments lost in a huge quantity of other fragments [1].

Consider a double stranded molecule that needs to be replicated. The molecule is flanked on both ends with specific sequences called primers. For the PCR process, an initial solution containing the required DNA molecule, the two primers, polymerase enzyme and nucleotides is prepared. The operation is then carried out in three steps:

- Denaturation of the double stranded molecule
- Priming or annealing the primers to the denatured strands
- Extension of the primers to form two copies of the double stranded molecule

Initially, the solution is heated to denaturize the target DNA molecule and then priming is carried out, during which the solution is slowly cooled down for the primers to anneal to the DNA single strands. The polymerase will then extend the primers using the nucleotides present in the solution to produce two identical DNA strands. The target molecule can also be a part of a longer DNA molecule. The process of PCR is depicted in Figure 2.12 [11].

Repeating the PCR process n times will lead to $2^n$ copies of the DNA molecule, thus increasing the copies exponentially. It is, however, important to note that the PCR process cannot be carried out without the knowledge of the primer sequences.
2.3 DNA computation models

Several abstract models of DNA computation have been proposed so far, most of which derive inspiration from Adleman’s experiment to solve the Hamiltonian path problem [1]. An abstract model of DNA computation does not consider any implementation issues and assumes that all computations take place upon a set of DNA strings.

For a given problem, a DNA algorithm takes the input encoded as a set of strings on which a series of DNA operations are performed. The output of the problem is also...
encoded as a string. Adleman’s unrestricted model which is based on his experiment to solve the Hamiltonian path problem is discussed in this section.

2.3.1 Adleman’s unrestricted model

Adleman’s unrestricted model is based on generation of the solution space through linear self assembly of DNA and filtering out the unnecessary fragments to obtain the final output.

**Adleman’s experiment**

Adleman’s experiment solves the Hamiltonian Path Problem (HPP) for a directed graph with seven vertices using tools from molecular biology [12]. Hamiltonian path problem can be formulated as follows:

Let $G(V,E)$ be a directed graph with $V$ vertices and $E$ edges. For selected input vertex $v_{in}$ and output vertex $v_{out}$, graph $G$ is said to have a Hamiltonian path if the path from $v_{in}$ to $v_{out}$ contains every vertex only once. The Hamiltonian path problem is to find such a path if one exists.

The directed graph considered by Adleman is shown in Figure 2.13 [12]. In this graph, the input is selected to be vertex 0 and the output is selected to be vertex 6. The path $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$ is Hamiltonian since each vertex is contained only once in the path.
HPP is an NP-complete problem and hence cannot be solved easily using the traditional computing techniques. Also, the heuristics that are used to solve the problem in a computationally feasible time do not always result in an optimum solution. DNA computing technique used by Adleman gives an optimal solution to this problem by performing an exhaustive search of the entire solution space, which is possible due to the vast parallelism inherent in the DNA molecules. Adleman also makes use of the complementarity of the DNA molecules to carry out the computation. The following non-deterministic algorithm has been used in Adleman’s experiment to solve HPP. Each step of the algorithm is also explained later.

- **Input:** A directed graph with $n$ vertices among which $v_{in}$ and $v_{out}$ are selected
- **Step 1:** All possible paths are generated. This comprises the entire solution space
- **Step 2:** All paths that do not start with $v_{in}$ and end with $v_{out}$ are discarded
- **Step 3:** All paths that do not include all the $n$ vertices are discarded
- **Step 4:** All paths that do not contain all the vertices exactly once are discarded
• **Output**: Hamiltonian path exists from $v_{in}$ to $v_{out}$ if any path remains. Else no Hamiltonian path

**Input**

The input to the problem is a set of DNA oligonucleotides (also called oligos in short) that encode the vertices and the edges of the graph. Each vertex $v_i$ of the graph is encoded by a random DNA oligo of length 20. For example, $v_2$ and $v_3$ are encoded as follows:

$v_2 = \text{TATCGGATCGGTATATCCGA}$

$v_3 = \text{GCTATTCGAGCTTAAAGCTA}$

Let $v_i$ be denoted as $v_i = v_i'v_i''$ where $v_i'$ is the first half of the sequence and $v_i''$ is the second half of the sequence. Then an edge $e_{i,j}$ from vertex $v_i$ to $v_j$ is encoded as $e_{i,j} = h(v_i''v_j')$, where $h(s)$ denotes the Watson-Crick complement of the DNA oligos. Hence the encoding of the edge $e_{2,3}$ is represented as follows:

$e_{2,3} = \text{CATATAGGCTCGATAAGCTC}$

**Step 1**

The first step of the experiment is to generate all possible paths. This is done by mixing all the encoded DNA oligos and allowing for the ligation reaction to happen. The encoding of the edge molecules is such that they link the two compatible vertex molecules. The resulting DNA sequences could be viewed as random paths through the graph.
**Step 2**

For step 2, to determine all the valid paths, PCR operation is performed with the $v_{in}$ and $v_{out}$ DNA oligos as the primers.

This results in the amplification of only those paths that start in $v_{in}$ and end in $v_{out}$.

**Step 3 and Step 4**

The resulting DNA sequences are then subjected to a series of filtering steps to filter out the unnecessary paths. To find DNA sequences containing only $n$ vertices, gel-electrophoresis is carried out and sequences of length $n*20$ (length of each DNA encoding is 20) are separated out. Then the DNA sequences containing each DNA oligo $v_i$ are extracted using affinity purification process. If any DNA sequence is extracted successfully, that represents the Hamiltonian path.

**Abstract formulation of Adleman’s experiment**

Some of the terms used for the formulation are defined below.

- **Test Tube**: Test tube is considered as a set of DNA strings.
- **Ligate**: Ligation of single stranded DNA sequences to form dsDNA molecules.
- **Merge**: Contents of two test tubes $T1$ and $T2$ are combined.
- **Length Separate**: Given a test tube $T$, and length $L$, a test tube $(T, \leq L)$ containing all DNA sequences of length at most $L$ is produced.
- **Extract**: Given a test tube $T$ and string $w$, the set $+(T,w)$ consisting of all strands that contain $w$ is produced.
- **Detect**: If a test tube $T$ contains at least one DNA molecule return true else return false.
Amplify: Given a test tube $T$ and primers $p_1$ and $p_2$, two copies of $(T,p_1,p_2)$ are produced.

Using this notation, a DNA algorithm [7] for the Adleman’s experiment is given below.

**Input ($T$) // initial solution space generated randomly through ligation**

$$T \leftarrow (T,v_{in},v_{out}) \quad /\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!}\] assignment

2.4 Errors in DNA computation

Biological operations are prone to several errors in spite of the well-defined procedures to carry out these operations and hence DNA computation is not error free [13]. The possibility of DNA computation as a future technology is limited due to these inherent biological errors, which affect the reliability of the technique. Errors associated with the common DNA operations and possible solutions to reduce their impact are discussed in this section.

2.4.1 Annealing

This is one of the most frequently used DNA operations. In annealing, two complementary DNA strands pair up to form a double stranded DNA molecule. Several errors are possible during annealing reactions which include mismatches in the bases leading to bubbles within the molecule and formation of hairpin structures due to...
folding of the molecules. These mismatch errors can be avoided by choosing proper DNA encodings for the data. The design of such DNA encodings is discussed in chapter 5.

2.4.2 Ligation

Ligation reaction, in which two DNA molecules are joined together in the presence of the enzyme ligase, is a relatively error free process.

2.4.3 PCR

PCR is a highly sensitive and very frequently used process in DNA computation and is generally assumed to be mostly error-free. This however depends on the problem size. The polymerase enzyme that is used for extending the primers in the PCR reaction sometimes makes wrong base-pairings. This is called misincorporation. Such misincorporation errors are in the range of one misincorporation per one hundred thousand bases, which is very small and can be neglected for short DNA sequences.

Another source of error in PCR is usage of a large quantity of primers, which results in interaction among the primers and generation of DNA strands of unexpected lengths. This error can be avoided by using only the required quantity of the primers for the action being performed.
Chapter 3
DNA cryptography

DNA cryptography is a relatively new field in which DNA molecules act as information carriers and DNA operations are used for the encryption of the information [14]. The Watson-Crick complementarity of the nucleotides that make up the DNA molecules leads to self assembly structures which facilitate the implementation of parallel computations. Due to this vast parallelism along with high information storage capacity, DNA can be considered to have a high cryptographic strength [14].

Though not very mature in terms of theory and realization, DNA cryptography is a fast growing interdisciplinary field, in which successful attempts were made to break the traditional cryptographic algorithms and also several algorithms were proposed for data encryption and steganography. This chapter deals with the basics of cryptography, gives a brief outline of DNA cryptography and explains an existing DNA cryptography algorithm based on the PCR technique.

3.1 Overview

In brief, cryptography is the science of information security. It is one of the oldest and most widely used fields due to the fact that secure communications has always been of great importance. With the advent of computers and increase in internet usage, the significance of this field has greatly increased. Cryptography allows the secure transmission of data over insecure channels such as internet and has applications ranging from secure commerce and payments to private communications and protecting passwords [15].
While cryptography aims at secure exchange of information, cryptanalysis is the science of analyzing and breaking the cryptographic methods of secure communication. A cipher is considered to be secure if it can be shown that there exists no efficient method (other than the brute-force method, which is extremely time consuming) to break the cipher. One-time pad cipher is one such cipher that has been proved to be unbreakable given that the one-time pad is used only once.

Classic cryptography is essentially encryption of messages to make them undecipherable for unintended receivers. This dates back to the early Egyptian era when hieroglyphs were used in inscriptions. However, the classic ciphers were extremely vulnerable to cryptanalysis and could be easily broken by frequency analysis techniques, etc. Modern cryptography, due to the advancements in mathematics and computational ability, employs much more complex cryptographic algorithms. It overlaps with the disciplines of mathematics, computer science and engineering and makes use of computationally hard problems in mathematics for data encryption [16].

For any application, authentication, privacy, integrity and non-repudiation form some of the security requirements [14]. In modern cryptography, these requirements are satisfied by three types of cryptographic schemes, namely symmetric key cryptography, public key cryptography and hash functions. Some of the highly secure cryptosystems that are used commonly like RSA, AES, ECC etc., are a result of extensive work in the cryptography and cryptanalysis fields.

The typical cryptographic scenario is explained using Alice and Bob, the archetypal characters in cryptography. Alice (Sender) wants to send some information securely to Bob (Receiver) in the presence of some third party adversary. The original message
that Alice wants to send is called the plain text or clear text. Plain text is usually in ordinary language and can be understood by all. Alice encrypts this plaintext message using any of the cryptographic algorithms to make it undecipherable by the third party adversary. This encrypted message which can be read only with the help of some special knowledge is called the cipher text. The special knowledge is called the encryption key.

The cipher text can be sent to Bob over any insecure channel. Bob, using his special knowledge, called the decryption key, decrypts the cipher text obtained using some cryptographic algorithm. Decryption is the reverse process of encryption to convert the cipher text to the original message using the decryption key. Since only Alice and Bob possess the keys, the cipher text will be unreadable to the third party adversary. This process of encryption and decryption is shown in Figure 3.1 [17] [18].

![Figure 3.1: Encryption and decryption process](image)

In general, cryptographic schemes are usually based on difficult problems to which a known efficient solution does not exist. The modern cryptographic schemes are designed around hard mathematical problems such as prime factorization and elliptic curve problem. [19]. Another well-known subfield of cryptography is the relatively new field of quantum cryptography which is based on the quantum mechanical effects [19].
Finally, the field of DNA cryptography is also designed around biologically difficult problems such as polymerase chain reaction etc [20]. Various aspects of DNA cryptography are explained in the following sections.

### 3.2 Comparison of DNA cryptography to traditional cryptography

As mentioned earlier, DNA is potentially a powerful tool for use in cryptographic systems. The plaintext messages of DNA can be encoded using the DNA alphabet \{A, G, T, C\} [20] or using short oligonucleotides. Original DNA sequences obtained from biological sources can also be recoded using different base pairs for subsequent processing [19]. Traditional cryptography is typically implemented on silicon and uses the algorithms from modern cryptosystems. A comparison between traditional and DNA cryptographic techniques has been done in [19] and is briefly described here.

The aspects of the comparison of cryptographic algorithms in DNA cryptography and traditional cryptography are as follows.

1. **Security** provided by the technique: This allows us to compare how secure the cryptographic techniques are.
2. **Time taken** to process the technique: Processing time is an important parameter to decide the practicality of the technique.
3. **Storage capacity** of the medium used: This parameter allows us to understand the amount of information that can be handled by the technique.
4. **Stability** of the results: The stability of results indicates that the encryption and decryption by the cryptographic techniques always provide the same results.
Table 3.1 [19] summarizes the comparison results of the cryptographic techniques in both these fields.

<table>
<thead>
<tr>
<th>Traditional cryptography</th>
<th>Security</th>
<th>Time complexity</th>
<th>Storage medium</th>
<th>Storage capacity</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>One fold: Mathematically based</td>
<td>≥ few seconds</td>
<td>Computer (silicon) chips</td>
<td>1 gram of silicon chip carries 16MB</td>
<td>Dependent on implementation environment</td>
</tr>
<tr>
<td>DNA cryptography</td>
<td>Twofold: Mathematically and biologically based</td>
<td>≥ few hours</td>
<td>DNA strands</td>
<td>1 gram of DNA carries 10^8 TB</td>
<td>Dependent on environmental conditions</td>
</tr>
</tbody>
</table>

**Table 3.1: Comparison between modern and DNA cryptographic techniques [19]**

It can be observed from Table 3.1 [19] that the security provided by DNA cryptography is twofold as opposed to the modern cryptography that provides only a single level of security due to the fact that DNA cryptographic schemes are based on both computational and biological difficulties. Also, the DNA strands as storage medium provide an unparalleled storage capacity which makes the field of DNA cryptography attractive for further research. However, the DNA techniques are extremely time consuming compared to that of the traditional techniques. While it takes hours to implement the biological processes like PCR, it takes only few seconds to implement the modern cryptographic algorithms. The stability of traditional cryptographic methods depend on the implementation conditions such as the platform of implementation and the language used for encryption algorithms. The stability of results in DNA cryptography is environment dependent and is subject to fluctuations in temperature and pH.
Though DNA offers specific advantages over traditional cryptography, practical implementations of these encryption algorithms require a lot of resources and time. Hence simple and efficient algorithms are necessary [14]. A lot of research work is required both experimentally and theoretically to realize simple algorithms and use them for practical purposes. That being said, one has to realize that at the current stage, DNA cryptography cannot totally replace the traditional cryptographic techniques. However, hybrid cryptosystems involving both the fields can be developed to improve the security level and to be able to handle huge amounts of information.

3.3 Previous work in DNA cryptography

Preliminary work in the field of DNA cryptanalysis, where some of the famous traditional cryptographic algorithms like RSA and DES [9] have been broken using DNA computing techniques, inspired researchers to apply DNA computing to encrypt data.

Several DNA based algorithms have been proposed and implemented for encryption of data and images and also for steganography purposes. While some of the DNA based algorithms proposed have made use of the difficult DNA operation called Polymerase Chain Reaction (PCR) [20] [21], some algorithms made use of DNA chip technology to implement their cryptosystems [22] [23]. Tile assembly model of DNA has also been used effectively to execute one-time pad cryptosystems successfully [22], [24]. While most of these algorithms use DNA sequences to store information and process them, there are a few algorithms that only use the concept of DNA sequences and operations to implement them in silicon [25]. Some image encryption algorithms based on DNA are also found in literature [26], [27]. DNA microdot [28], a steganography method for
secret writing purpose, has also been proposed. PCR based algorithm is discussed below.

### 3.3.1 Cryptography based on PCR of DNA

Cui et al. [20] have proposed a hybrid encryption scheme using the PCR amplification process, DNA digital coding and traditional cryptography. Prabhu et al. [21] have designed a bi-serial encryption algorithm based on PCR amplification which provides two levels of security.

PCR (Polymerase Chain Reaction) is an amplification and quantification process of DNA. The implementation of PCR has been discussed in Chapter 2. However, one property of PCR which makes it useful for cryptographic purposes is that the amplification of the DNA sequences cannot be carried out unless the exact sequence of the primers is known. How this property of PCR can be made use of in encryption of data is discussed below.

The PCR process consists of 1) Hybridization of forward and reverse primer sequences to the actual DNA sequence 2) Extension of the primers along the DNA sequence to create a copy. For the purpose of encryption, the primers are used as the encryption and decryption keys.

**Key generation**

In the typical cryptographic scenario where Alice wants to send some information to Bob, the keys, i.e., the DNA primers, are generated and exchanged through a secure communication channel by Alice and Bob. As an example consider the two primers to be encoded as follows:
\[ P1 = 5'\text{-ATGTGTATCTGGATCCTAGT-3'} \]

\[ P2 = 5'\text{-GTGATGATGATCCGCTAGCT-3'} \]

**Encryption**

For encryption, the original data is subject to some preprocessing which is then converted to a DNA cipher text using DNA digital coding techniques by Alice. The DNA digital coding is defined in [20] as follows:

\[ C = 00, T = 01, A = 10, G = 11 \]

If the binary text that needs to be encoded is “10010111”, then its DNA encoding would be “ATTG”.

The DNA coded message is then flanked by the predecided forward and reverse PCR primers and then by long dummy DNA sequences on either side of the primers and is sent to Bob over any channel of communication. Hence, in the above example, the encrypted message would be as follows:

\[ \ldots\text{GTGACT} \text{ATGTGTATCTGGATCTAGT} \text{ATTG} \text{GTGATGATGATCCGCTAGCTTCTAGC}\ldots \]

The DNA sequence highlighted in bold is the original message, flanked by primers shown in italics and then flanked by long random DNA sequences.

**Decryption**

The decryption process involves the retrieval of the DNA cipher text by performing PCR amplification on the long DNA sequence that Bob received using the primer keys. Once the DNA cipher text is obtained, it can be converted to the original text after the required post processing is done. In the above example, if the two primer keys P1 and
P2 are known, then the message can be extracted easily from the long DNA sequence. This cryptographic process is shown in Figure 3.2 [20].

![Figure 3.2: PCR based cryptosystem [20]](image)

In [20], the preprocessing involved application of traditional cryptography methods such as RSA or AES to the original text, converting the obtained cipher text into binary and applying DNA digital coding technique to convert it from binary to a DNA sequence. In [21], the preprocessing of data required converting the original message into binary format and XOR-ing it to obtain high compression. The resulting data is converted to a DNA sequence using the DNA digital coding technique.
Chapter 4
DNA self-assembly based cryptographic algorithm

In this chapter, a DNA cryptographic algorithm based on linear self assembly of DNA has been proposed. As mentioned before, DNA cryptography only provides an additional security level to the modern cryptosystems and acts as a bridge between modern cryptography and the new technologies. It has to be noted that techniques for breaking complex cryptosystems such as RSA have been proposed based on DNA computing. Hence, an additional level of security becomes necessary.

The DNA encryption algorithm presented here is applied on an already processed plain text using any of the modern cryptosystems. The encryption algorithm is based on the One-Time Pad (OTP) cryptography algorithm. The security of this algorithm is highly dependent on the randomness of the key which, when ensured, is known to be unbreakable [24].

Given a random key (K) of length equal to the length of the message (M), the encryption can be done as follows. Let the cipher text be denoted by C:

\[ C = M \oplus K \]
4.1 One-time pad generation

First, a large library of one-time pads is constructed using the DNA self assembly process. A single DNA sequence from this library is taken as the random key for encryption/decryption. These one-time-pads are assumed to be constructed in secret, and they are shared in advance by both the sender and receiver of the secret message. This assumption requires initial communication of the one-time pad between sender and receiver, which is facilitated by the compact nature of DNA.

4.1.1 DNA coding of binary numbers

Let $A = a_1, a_2, \ldots, a_n$ represent a binary string of length $n$ where $a_i \in \{0, 1\}$. Let each $a_i(0)$ or $a_i(1)$ be represented by a unique DNA sequence ($D_i(0)$ or $D_i(1)$) of length $L$. While ‘i’ indicates the position of the bit, the number in the bracket indicates the value of the bit. It is assumed that the position encodings of DNA are available to both sender and receiver parties.

**Example:**

Let $n=4$;

Let the DNA encodings for the positions be assumed as follows. These encodings are selected randomly.

$$a_1(0) = 5'-\text{TAGC}-3'; \ a_2(0) = 5'-\text{TTAG}-3'; \ a_3(0) = 5'-\text{GGAC}-3'; \ a_4(0) = 5'-\text{GATC}-3'$$

$$a_1(1) = 5'-\text{GCTA}-3'; \ a_2(1) = 5'-\text{TCAG}-3'; \ a_3(1) = 5'-\text{GTAT}-3'; \ a_4(1) = 5'-\text{GAGT}-3'$$

Hence the binary number $A=1001$ can be represented using DNA sequences as follows:
4.1.2 Implementation using DNA operations

Ligation reaction can be used to generate a library of one-time pads through linear self assembly of the encoded DNA oligonucleotides. The process is given below:

**Step 1**

Prepare two test tubes $T_U$ and $T_L$ such that,

$T_U = \{D_k(0) D_{k+1}(0), D_k(0) D_{k+1}(1), D_k(1) D_{k+1}(0), D_k(1) D_{k+1}(1), \text{ where } k \in \{1,2,..n\}\}$

$T_L = \{(D_k(0) D_{k+1}(0))', (D_k(0) D_{k+1}(1))', (D_k(1) D_{k+1}(0))', (D_k(1) D_{k+1}(1))'\}, \text{ where } k \in \{2,..n-1\}$

**Step 2**

Test tubes $T_U$ and $T_L$ are merged together. This results in self assembly of the oligonucleotides in a random way.

**Step 3**

The DNA strands of length $(n*L)$ represent binary numbers of length $n$ and hence can be separated using gel electrophoresis. These DNA strands form the one-time pad library.

**Example**: Using the DNA encodings from above example where $n=4$;

$T_U = \{D_1(0) D_2(0), D_1(0) D_2(1), D_1(1) D_2(0), D_1(1) D_2(1),$

$D_3(0) D_4(0), D_3(0) D_4(1), D_3(1) D_4(0), D_3(1) D_4(1)\}$

$T_L = \{(D_2(0) D_3(0))', (D_2(0) D_3(1))', (D_2(1) D_3(0))', (D_2(1) D_3(1))'\}$
Mixing of test tubes and later separating the DNA strands based on length will yield 4-digit binary numbers. Few such DNA strands are shown below.

\[
\begin{align*}
D_1(0) & \ D_2(1) \ D_3(1) \ D_4(0) \Rightarrow \ 5'-TAGCTCAGGTATGCAT-3' \Rightarrow 0110 \\
D_1(1) & \ D_2(0) \ D_3(0) \ D_4(0) \Rightarrow \ 5'-GCTATTAG\ GGACGCAT-3' \Rightarrow 1000 \\
D_1(1) & \ D_2(1) \ D_3(1) \ D_4(0) \Rightarrow \ 5'-GCTATCAGGTATGCAT-3' \Rightarrow 1110
\end{align*}
\]

For the encryption, one of the onetime pads can be randomly selected and sent securely to the receiver using DNA microdot [28].

4.2 Encryption

The preprocessed plain text (M) is converted to binary representation. A onetime pad of length equal to or greater than ‘M’ is selected randomly as discussed above. Let \( M = m_1m_2...m_n \). Each bit of M can be encoded by DNA using Watson-Crick complementary DNA sequences of length L.

To perform the XOR operation, the one-time pad DNA sequence is mixed with the DNA sequences of M. The strands from M that are complementary to the subsequences of the onetime pad get annealed to it. The remaining single sequences represent the 1 bits of the cipher text. The binary representation of the cipher text can be generated using those DNA single strands.

**Example:**

Let \( M=1001 \); Let K (one-time Pad) = 1110; \( C=M\oplus K = 0111 \)

DNA encodings (Using the above example):

\[
K= \ 5'-GCTATCAGGTATGCAT-3'
\]
When K and M are mixed in a test tube, the following strands will be formed:

\[ \begin{array}{c c c c c c c}
5' & AGGC & GCTA & TCAG & GTAT & GCAT & AGGC & \ldots \\
| & | & | & | & | & | \ldots \\
CGAT & | & | & | & | & | \\
\end{array} \]

Hence, the 3 rightmost bits in the cipher text are 1’s. The binary representation will be: 0111

4.3 Decryption

Similar process is employed for the decryption. The bits of the cipher text are encoded by DNA sequences and annealed with the one-time pad. This results in an XOR operation between K and C and the resulting single strands of DNA can be used to generate the binary representation of M.

4.4 Implementation issues

Ideally, for an oligonucleotide of length L, \(4^L\) DNA sequences are possible. But the number of DNA sequences that can be used are limited to less than \(4^L\) due to hybridization stringencies. The size of the plaintext that can be encrypted depends on the number of DNA sequences that can be used to encode it. If the length of the message is longer than the one-time pad that can be constructed, then more than one one-time pad should be used.

For example, for a length L=20, around \(10^{12}\) unique DNA sequences can be generated. But hybridization constraints limit the number of DNA sequences that can be used to prevent the occurrence of false hybridizations. Even if around \(10^6\) unique DNA
sequences can be identified, data of around $10^6$ bits can be encrypted. The hybridization constraints on DNA encodings and design of good DNA encodings are discussed in chapter 5.

4.5 Evaluation of the cryptographic algorithms

An evaluation of the DNA cryptographic algorithm proposed in this chapter and the PCR based cryptography algorithm discussed in the previous chapter is presented in this section. The following characteristics of the cryptographic algorithms have been identified to be used as metrics for evaluating and comparing the various algorithms.

- Maximum data size: Since the input data is encoded in a different way for each type of the cryptographic algorithm, this can be used as a metric to evaluate the feasibility of the algorithm for large data sizes.

- Number of biological operations used: Since the biological operations consume quite some time and are also error prone, it is important to keep the number of biological operations to a minimum.

- Possible error rate during encryption/decryption: Since DNA operations are prone to errors, an estimate of the error rates that could be introduced in the encryption / decryption process is an important parameter for evaluating the cryptographic algorithm.

**PCR based cryptography**

DNA molecules are known to store huge amount of information. A typical human DNA contains more than a billion base pairs, which can store more than 1 GB of data. Since the binary message is DNA digital encoded in this technique, maximum data size could be as high as 1GB or even more. The encryption process requires only one
DNA operation i.e., synthesis of the encoded data with the primers flanked on either side. The decryption process requires PCR operation and sequencing of the extracted DNA sequences. The PCR process is mostly error free. However, selection of primers is very crucial for the PCR operation. If primers are not selected properly, there is a chance that the primers will anneal at a wrong hybridization site thus producing incorrect DNA plaintext sequences.

**Linear self assembly based cryptography**

The maximum data size for this type of cryptography is limited due to the hybridization errors. For an oligonucleotide length $L$, if around $10^6$ unique DNA encodings can be generated that satisfy the hybridization constraints, then only around $10^6$ bits of data can be encrypted.

The DNA operations involved in key generation are synthesis, ligation and gel electrophoresis all of which can be assumed to be relatively error free and in encryption/decryption hybridization reaction is carried out which also can be assumed to be error free if good encodings of DNA sequences are used.
Chapter 5

Generating reliable DNA sequences

5.1 Overview

DNA computing relies heavily on the hybridization of DNA sequences during the computation process and its reliability depends on the ability to control these hybridization reactions. Undesirable hybridizations between the DNA sequences are called false hybridizations and may result in inappropriate computing results. Hence, it becomes very important to use reliable DNA sequences during the computation process and the design of such DNA sequences is central to the field of DNA computation. This chapter deals with the generation of good DNA encodings for reliable DNA computation.

Research has been carried out in similar lines in the past to solve the DNA encoding problem by proposing various design constraints and different algorithms to approach the problem. Hamming distance constraints have been used for generating reliable DNA sequences [31] [32]. But, Hamming distance does not measure the frame-shift errors which also must be considered. A new metric called H-measure has been proposed in [33] that also takes into account the frame-shift errors, which cannot be estimated by the Hamming distance constraints. This is described below. Algorithms using the Hamming distance [31], [32] and the H-measure [34] are also discussed below.
In this chapter, implementation of simulated annealing algorithm for the DNA encoding problem is discussed and compared to the previous work that has been done in this area.

5.2 Hybridization errors

False hybridization essentially means that two unintended DNA oligonucleotides hybridize during the computation process. Due to accumulation of these false hybridization errors in the computation process, the final result obtained might be far from the intended result.

These false hybridizations result in two types of errors called false positives and false negatives [30]. False positives are unnecessary DNA sequences which get extracted along with the required DNA sequences. False negatives are those DNA sequences that are wrongly classified as the unnecessary DNA sequences.

False negatives do not pose a serious threat to the computation process as the extraction process can be repeated to retrieve such DNA strands. However, false positives lead to an erroneous result to the problem being computed and hence, they need to be minimized by selecting good DNA encodings.

Some of the different kinds of hybridization are described below.

**Complete hybridization**

Complete hybridization occurs when two sequences are completely complementary to each other. This type of hybridization is the desirable one in DNA computation. This is shown pictorially in Figure 5.1
Hybridization with mismatches

Two sequences which are nearly complementary to each other (but not completely) pair up together which results in mismatches or bubbles in the DNA structure. More details on hybridization mismatches can be found in [35]. This is shown in Figure 5.2.

Translocation hybridization

This kind of hybridization occurs between oligonucleotides that are shifted with respect to one another. In Figure 5.3, the lower DNA sequence is shifted with respect to the upper sequence to hybridize at a translocation. This kind of hybridization gives rise to errors known as frame shift errors [33] which need to be minimized through selection of good sequences.
**Self hybridization**

An oligonucleotide which has a part of it that is complementary to itself can fold back upon itself to self-hybridize and form a secondary structure called DNA hairpin. This kind of hybridization does not allow the oligonucleotide to further participate in the computation process and is also undesirable. Formation of a hairpin structure due to self hybridization is shown in Figure 5.4.

![Figure 5.4: Hairpin structure formation by self hybridization of the DNA sequence](image)
5.3 DNA encoding problem

The design of reliable DNA sequences has been experimentally studied earlier by several researchers [29] and such empirical information can be used to develop an objective function that is subject to multiple constraints. The generation of consistent DNA sequences is hence considered to be a multi-objective optimization problem and is termed as the DNA encoding problem [33].

The DNA encoding problem entails the design of the DNA sequences in such a way that during a hybridization reaction, the sequence anneals to its own designated complement sequence and prevents false hybridizations.

For a given length $L$, there would be $4^L$ possible DNA sequences that could be used for the computation process. Selecting ‘n’ random sequences ($n<4^L$) for a particular computation might eventually lead to hybridization errors and a consequent invalid result to the computation problem. However, selecting the ‘n’ sequences after the optimization of the combinatorial constraints could almost guarantee an error free computation process. For small ‘n’, the DNA encoding problem is relatively easy to solve. But with increasing ‘n’, generation of DNA sequences, which satisfy all the constraints, within a reasonable time, may be difficult. This problem was shown to be NP-complete [33].

The DNA encoding problem can be formulated as follows [32]:

Given the alphabet $\Sigma = \{A, G, T, C\}$ for encoding the DNA sequence and a set $Z$ of all possible permutations of DNA sequences over $\Sigma$, such that $|Z|=4^L$, search for the set $W$ in $Z$ such that, $\forall (x_i,x_j) \in W$, $\tau(x_i,x_j)>k$, where $k$ is a positive integer and $\tau$ is the
expected criterion of evaluating the encoding i.e., the encodings should satisfy all the
given combinatorial constraints.

In this work, the Simulated Annealing (SA) algorithm for the DNA encoding problem
has been used. The cost function is subject to several combinatorial constraints and is
formulated as a minimization problem. The following sections describe more about the
design considerations and implementation of the algorithm.

5.4 DNA sequence design considerations

DNA sequences should ideally satisfy both combinatorial and thermodynamic
constraints. In this thesis, only combinatorial constraints have been considered to
generate the sequences. In this section, two metrics for estimating the hybridization
likelihood have been discussed first and then the cost function has been described.

5.4.1 Hamming distance constraint

If \( x_i \) and \( x_j \) are two DNA sequences, then Hamming distance \( H(x_i, x_j) \) is defined as the
number of positions at which the corresponding bases are different in \( x_i \) and \( x_j \). For
example, let \( x_i = CATGTCGTA \) and \( x_j = GACACGTGC \). The letters in bold indicate
the similar bases at corresponding positions in both the sequences. Hence, Hamming
distance \( H(x_i, x_j) = 7 \).

For ‘n’ DNA sequences with length \( L \) and 5’-3’ directionality, the Hamming distance
constraint is specified as follows:

\[
\begin{align*}
\hat{f}_{\text{hamming}} &= \min_{1 \leq j \leq n, j \neq i} H(x_i, x_j^{RC}) \\
\end{align*}
\]

Where, \( x^{RC} \) denotes the Watson-Crick complement of \( x \).
Sequences with maximum Hamming distance are required to reduce the occurrence of false hybridizations. For example consider two sequences \( x_i \) and \( x_j \).

\[
x_i = AG\underline{GT}TCATGGA
\]

\[
x_j = AGACCTCGTCAG \; x_j^{RC} = CT\underline{GAGG}AGGTCT
\]

Hence \( H(x_i, x_j^{RC}) = 10 \). The similar bases at respective positions are highlighted in bold. Sequences with high Hamming distance have less possibility to hybridize. In general, a Hamming distance greater than \( L/2 \) is desired, where \( L \) is the length of the DNA sequences [31].

### 5.4.2 H-Measure

Though Hamming distance constraint captures the possibility of partial hybridizations well, it falls short of estimation of the frame-shift errors. For example, consider the sequences from the previous example:

\[
x_i = AG\underline{GT}TCATGGA
\]

\[
x_j = AGACCTCGTCAG \; x_j^{RC} = CT\underline{GAGG}AGGTCT
\]

Though the Hamming distance is as high as 10, a translocation hybridization is possible between the two sequences giving rise to a frame shift error. This is shown in Figure 5.5.

![Figure 5.5: Translocation hybridization between two sequences with high Hamming distance](image)
Hence, Hamming distance constraint is inadequate to measure the hybridization repulsion. Garzon et al. have proposed a new metric for incorporating the frame-shift errors which is called H-measure [33], which is more appropriate to measure the hybridization likelihood and is given below.

For two oligonucleotides \( x_i \) and \( x_j \) of length \( L \), the H-measure is defined as follows:

\[
|x_i x_j| := \min_{-L \leq k \leq L} H(x_i, \sigma_k(x_j^{RC}))
\]

Where, \( x_j^{RC} \) denotes the Watson-Crick complement of \( x \), \( k \) denotes the number of shifts and \( \sigma^k \) denotes right shift for \( k > 0 \) and left shift for \( k < 0 \). The H-measure takes the minimum Hamming distance between \( x_i \) and all the shifted sequences of \( x_j \). Thus a large H-measure indicates that two oligonucleotides \( x_i \) and \( x_j \) will not contain any complementary segments and hence will not hybridize under any physico-chemical conditions [33].

5.4.3 Cost function

The various terms of the cost function developed in this work are based on those defined in [29] and are described below.

Let ‘\( n \)’ be the required number of DNA sequences of length \( L \) and let a DNA sequence be denoted as \( x_j \) \((1 \leq i \leq n)\). For two sequences \( x_i \) and \( x_j \), \( H_S(x_i, x_j) \) is defined to be the number of bases that are same at corresponding positions. The different terms of the cost function considered in this work are described below.

**Modified H-measure**

H-measure metric has already been discussed above. H-measure term ensures that there is no inappropriate complete, partial or translocation hybridization between any
two sequences. The original definition has been modified to make it appropriate for the minimization problem and is given below.

\[ f_H = \sum_i \sum_{j, j > i} \max_{-L < k < L} \{H_S(x_i, \sigma^k(x_j^{RC}))\} \]

A lower H-measure indicates lesser similarity between the sequences and hence lesser chances of hybridization. Hence, the optimization involves minimizing the maximum similarity between the shifted sequences.

**Self-measure**

The self H-measure term measures the occurrence of secondary structure formation and is defined as follows.

\[ f_{self} = \sum_i \max_{-L < k < L} \{H_S(x_i, \sigma^k(x_i^{RC}))\} \]

**Similarity**

Two DNA sequences containing a same subsequence might mishybridize during the computation process and is highly undesirable and hence this evaluation term is considered.

\[ f_{sim} = \sum_i \sum_{j, j > i} \max_{-L < k < L} \{H_S(x_i, \sigma^k(x_j))\} \]
5.4.3 Combinatorial constraints for the DNA encoding problem

- **H-measure**: The modified H-measure should be minimized between all DNA sequences to avoid the false hybridizations.

- **GC content**: GC content of a DNA sequence is defined as the total number of G and C bases in it. Since the G-C bond is stronger than that of A-T, it is preferred to have the same or more number of G and C bases as that of A and T bases. In this work, it is considered that the GC content of the DNA sequences should be at least 50% i.e., if L is the length of a DNA sequence, GC content should be at least L/2. Also, the GC content of all the sequences is also required to be uniform.

- **Continuity**: A DNA sequence becomes unstable if it contains a single base continuously in its sequences and hence the reaction becomes uncontrollable. Hence, it is highly desirable that no base appears continuously more than twice in the sequence. For example, the sequence AGGGGGTCTACG is an undesired sequence, while AGGTCTCCGTATAGT is acceptable.

5.5 Simulated Annealing (SA) approach

5.5.1 Description

Simulated annealing algorithm is a stochastic iterative algorithm that can be used for multi-objective optimization problems and has been used for encoding reliable DNA sequences in this work.
SA draws inspiration from the annealing of metals in which the metals are heated to a very high temperature and then cooled down slowly such that the atoms find a final configuration with low internal energy than the starting configuration.

The algorithm starts with a random solution set and a high temperature $T_0$. $\alpha$, $\beta$, $M$ and maxtime are other parameters that are needed to control the progress of the algorithm. $\alpha$ determines the rate at which temperature is reduced while, $M$ and $\beta$ determine the cooling schedule. The iterations are stopped when maxtime is reached.

The main idea of SA algorithm is to generate a new solution and accept it based on certain conditions. If the cost of new solution is less than that of the old one (for a minimization problem) then the new solution is accepted. Else, it is accepted on a probability basis. The iterations are carried out until max time is reached. When the temperature is high, solutions with higher cost than the current solution are accepted which allows the algorithm to move out of the local minimum. As the temperature goes down, solutions are not accepted on probability basis anymore and hence the algorithm converges to a global optimum. Pseudo code for the basic SA algorithm is given below.

```plaintext
SimulatedAnnealing (S, $T_0$, $\alpha$, $\beta$, $M$, maxtime)
{
    $T=T_0$; $S = S_0$; time=0;

    while( time< maxtime) {
        Metropolis(S,T,M);
        time = time + M;
        $T = \alpha \times T$;
        $M = \beta \times M$;
    }

    Output best solution;
}
```

47
Metropolis(S,T,M)
{
    while(M>0)
    {
        S<sub>new</sub> = newSolution(S);
        Δh = cost(S<sub>new</sub>) – cost(S);
        if ((Δh<0) or (random < e<sup>−Δh/T</sup>))
        {
            S = S<sub>new</sub>; // accept the new solution
        }
        M=M-1;
    }
}

For the DNA encoding algorithm, the initial solution set is created by randomly generating DNA sequences such that they satisfy the GC content and continuity constraints. A new solution is generated by replacing a sequence from the current solution set with a random DNA sequence that satisfies the GC content and continuity constraints. The cost of a solution is given by determining the value of the evaluation function that is taken as a weighted sum of the three evaluation terms discussed previously. The cost function is taken as follows.

\[ f = w_1 \times f_H + w_2 \times f_{self} + w_3 \times f_{sim} \]

Flow chart for SA algorithm that has been used to solve the DNA encoding problem is given in Figure 5.6.
Figure 5.6: Flow chart for the simulated annealing algorithm
5.5.2 Implementation

The simulated annealing algorithm to generate reliable DNA sequences has been implemented using Java. The hardware configuration and software tools used for simulation are given below.

**Hardware configuration**

The computer on which the simulation has been run had the following hardware configurations:

- 2.3 GHz Intel i3 processor
- 4GB of RAM
- Windows 7 operating system

**Software used**

The following software has been used for implementation of the java code.

- Java SDK 7
- Netbeans IDE

5.5.3 Results

For the implementation of the algorithm, the parameters chosen were $T = 300$, $\alpha = 0.4$, $\beta = 2$, $M = 200$ and $\text{maxtime} = 100000$. The fitness plot of the simulated annealing algorithm for the above parameters and number of DNA sequences $n = 14$ and length of each DNA sequence $L = 20$ is shown in Figure 5.7. We can see that the algorithm converges to a global optimum eventually when the temperature reaches zero.
The dependency of simulation time as a function of length of DNA sequences, L and number of DNA sequences to be generated ‘n’ has been observed. The other algorithms with which the SA approach is compared in the next section, do not specify the dependency of their simulation time on the length of DNA sequence and the number of DNA sequences.

For L = 20, time versus number ‘n’ of DNA sequences generated is plotted in Figure 5.8. We can see that there is a polynomial increase in the time as n increases due to the fact that for calculating the H-measure, for every DNA sequence in the solution set, we have to iterate over all the reverse complements of these DNA sequences.

For a fixed number of DNA sequences n = 15, a plot of time versus L, the length of DNA strands is generated and shown in Figure 5.9. The increase in time is linear since L determines the number of shifts in H-measure calculation.
Figure 5.8: Plot of time (sec) versus number of DNA sequences 'n' for L = 20

Figure 5.9: Plot of time (sec) versus length of DNA sequence (L) for 15 DNA sequences
The GC content for the following experiments has been chosen to be 50%. The required GC content can be set in the GC content constraint which will ensure that all the DNA sequences that are generated will have the given GC content. The GC content can be specified to be 50% or higher.

**Experiment 1**

A set of 14 sequences of length \( L=20 \) were generated using simulated annealing approach and are shown in Table 5.1. Since we have assured that any new DNA sequence that is generated satisfies the GC content constraint and the continuity constraint, the GC content of every DNA sequence is uniformly 50% i.e., GC content = 10. Also we can see that any base does not appear continuously more than twice. The DNA sequences generated also have a low H-measure, self H-measure and similarity values.

<table>
<thead>
<tr>
<th>DNA Sequence</th>
<th>Modified H-measure</th>
<th>Similarity</th>
<th>Self-measure</th>
<th>GC content</th>
<th>Continuity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GGTGGATGTGGTGACTAAG</td>
<td>109</td>
<td>101</td>
<td>6</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>CACTCCACCATCACCACCTT</td>
<td>108</td>
<td>90</td>
<td>6</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>GGAGAGGTCAGTAGACTAAC</td>
<td>101</td>
<td>103</td>
<td>8</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>CGTTATGGTGGATAGTTG</td>
<td>101</td>
<td>104</td>
<td>8</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>CTGTTGTGTTGTCGTCTAC</td>
<td>106</td>
<td>101</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>GACAAGAAGACGGAGCG</td>
<td>112</td>
<td>92</td>
<td>6</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>GGTGCGTGAATGGTTGAG</td>
<td>104</td>
<td>108</td>
<td>6</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>ATTACGAGCAACCGGAACGA</td>
<td>106</td>
<td>102</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>TGTTGTCTCTGTCTTTGG</td>
<td>102</td>
<td>104</td>
<td>6</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>TCTCAGCACCACCCACCA</td>
<td>113</td>
<td>97</td>
<td>6</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>GTGTATAGGCCATGGTGAC</td>
<td>98</td>
<td>111</td>
<td>8</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>TGCTGTGTGGTGTGTATTG</td>
<td>100</td>
<td>108</td>
<td>4</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>TCGGCACCTCTCTATTTCAC</td>
<td>105</td>
<td>95</td>
<td>8</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>ACATAGACACCTGCGCACAA</td>
<td>113</td>
<td>102</td>
<td>6</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

*Table 5.1: 14 reliable DNA sequences generated by SA approach*
To estimate the consistency of the values, the experiment was run 10 times and the standard deviation and mean values were calculated. The Table 5.2 shows the values from the 10 runs and the calculated standard deviation and mean values. It can be seen that the standard deviation is low and that the values obtained are pretty consistent.

<table>
<thead>
<tr>
<th>Run</th>
<th>Average modified H-measure</th>
<th>Average Similarity</th>
<th>Average self-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>103</td>
<td>101</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>106</td>
<td>99</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>105</td>
<td>100</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>101</td>
<td>105</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>104</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>105</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>105</td>
<td>101</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>98</td>
<td>107</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>103</td>
<td>102</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>106</td>
<td>101</td>
<td>8</td>
</tr>
<tr>
<td>Mean</td>
<td>102.7</td>
<td>102.5</td>
<td>7.1</td>
</tr>
</tbody>
</table>

Table 5.2: Mean and standard deviation of data points from 10 runs for n=14

**Experiment 2**

A set of 7 sequences of length L=20 were generated using simulated annealing approach and are shown in Table 5.3. Since it has been assured that any new DNA sequence that is generated satisfies the GC content constraint and the continuity constraint, the GC content of every DNA sequence is uniformly 50% i.e., GC content = 10. It can be seen that any base does not appear continuously more than twice. The DNA sequences generated also have a low H-measure, self H-measure and similarity values.
Table 5.3: 7 reliable DNA sequences generated by SA approach

To estimate the consistency of the values, the experiment was run 10 times and the standard deviation and mean values were calculated. The Table 5.4 shows the values from the 10 runs and the calculated standard deviation and mean values. It can be seen that the standard deviation is low and that the values obtained are pretty consistent.

<table>
<thead>
<tr>
<th>DNA Sequence</th>
<th>Modified H-measure</th>
<th>Similarity</th>
<th>Self-measure</th>
<th>GC content</th>
<th>Continuity</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAGGTGTTGTCTCTCGCTC</td>
<td>47</td>
<td>47</td>
<td>6</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>ACGGTACATGAGGATTCCGA</td>
<td>47</td>
<td>46</td>
<td>8</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>GGATGACTGTGTGAAGGAAG</td>
<td>48</td>
<td>43</td>
<td>8</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>GGAGAACACGAGACAGATC</td>
<td>54</td>
<td>40</td>
<td>8</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>TCTCCACACGCCTATTCACT</td>
<td>51</td>
<td>41</td>
<td>8</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>TGTATGCGTGTGTGACTTG</td>
<td>47</td>
<td>6</td>
<td>4</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>TACTCTCAGTGTGCCTAG</td>
<td>52</td>
<td>46</td>
<td>8</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.4: Mean and standard deviation of data points from 10 runs for n=7
5.6 Comparison with previous work

The DNA sequences generated by the simulated annealing algorithm are compared to those generated by a genetic algorithm [34], branch and cut [31] and an evolutionary algorithm [32]. A brief explanation of the work done in these papers is given first and then a comparison is made.

5.6.1 Genetic Algorithm (GA) approach

Genetic algorithms are search algorithms that provide an optimized set of solutions through simulated evolution. For a given set of initial solutions, a set of more optimized solutions are evolved through natural selection processes. This is achieved by replacing poor solutions in the solution set by better solutions which result from the crossover of two good solutions. Thus the beneficial traits from the two parent solutions are passed on to get a new solution with greater potential.

A fitness function is defined to evaluate how good a solution is in comparison to other solutions and can incorporate several evaluation terms. In general, a solution with higher value of fitness function is considered to better. The basic operations in genetic algorithm are natural selection, reproduction, crossover and mutation.

Fitness values are used for selecting solutions to be passed on to the next iteration. The natural selection process happens stochastically in such a way that solutions having a higher fitness value are more likely to be selected. Reproduction is simple to make a copy of the current solution. Crossover simulates the mating of two solutions by randomly selecting a crossover site and swapping the gene values to produce offspring. Mutation operation alters the gene value of some solution. This operation is
required to maintain diversity in the solution set and thus preventing the algorithm to converge prematurely.

The genetic algorithm is defined below:

1. Select a random population of required size \( n \).
2. Evaluate the population fitness
3. Repeat 3 to 8
   i. Select best individuals as parents from the population
   ii. Crossover the parents to produce offspring
   iii. Mutate one of the individuals in the population
   iv. Evaluate the population fitness
4. Until terminating condition met

The terminating condition for the genetic algorithm can be any of the following

- Number of generations.
- An individual satisfying the required criteria.
- The fitness of the best individual cannot be further improved.
- A combination of the above terminating conditions.

A genetic algorithm (GA) to solve the DNA encoding problem was proposed in [34]. The sequences were subject to all the combinatorial constraints discussed above, i.e., similarity, H-measure, hairpin formation, GC content and continuity constraints.

5.6.2 Branch and Cut Algorithm (BCA) approach

Branch and cut is a combinatorial optimization algorithm for solving linear programming problems and is a combination of branch and bound and cutting plane
methods. It essentially searches for a cutting plane at every node of the branch and bound tree.

The branch and cut algorithm to solve the DNA encoding problem was used in [31]. A quaternary tree of depth n, the length of a DNA sequence was established to generate the DNA sequences by quaternary branching.

A dead node is defined as a node that leads to sequences that contain forbidden subsequences. A depth-first traversal of the quaternary tree is carried out till it reaches the n\textsuperscript{th} level to find feasible DNA sequences that satisfy the combinatorial constraints. If a dead node is hit during the traversal, backtracking is done to find another path downwards.

The DNA encodings generated by this method were ensured to satisfy the GC content constraint which limits the GC content between 40%-60%, forbidden subsequence constraint and Hamming distance constraint which restricts the sequences to have a Hamming distance of at least n/2.

5.6.3 Evolutionary Algorithm (EA) approach

As the name suggests, evolutionary algorithms are stochastic search algorithms that are based on the natural evolutionary processes. Evolutionary algorithm proceeds in a manner similar to the genetic algorithm (which can be considered as a type of EA). As the algorithm proceeds, evolution of population happens based on survival of the fittest strategy rather than natural adaption of the individuals and hence the final population is more optimal than the starting population.

Artificial Immune algorithm, a type of EA is used in [32] to solve the DNA encoding problem. This algorithm is based on the immune system functioning in the body
where antigens are the objective functions and the constraints and antibodies are the solution set. The algorithm is similar to that of GA and can be found in detail in [32].

The constraints that were used for the DNA encoding problem are the Hamming distance constraints, GC content and continuity constraints.

5.6.4 Comparison results

The results generated by simulated annealing approach are compared with those obtained from BCA [31], GA [34] and EA [32]. For the sequences reported in these papers, H-measure, self H-measure, similarity, GC content and continuity values have been calculated and then compared to the values obtained in this work.

A set of 14 DNA sequences with \( L = 20 \) have been reported in [31]. Table 5.5 shows the DNA sequences from [31] and their corresponding evaluation function values.

<table>
<thead>
<tr>
<th>DNA Sequence</th>
<th>H-measure</th>
<th>Self H-measure</th>
<th>Similarity</th>
<th>GC content</th>
<th>Continuity</th>
</tr>
</thead>
<tbody>
<tr>
<td>AACAAGCCACCACCAACAC</td>
<td>107</td>
<td>2</td>
<td>106</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>AACAATGGTGTTGGGTTAGGA</td>
<td>115</td>
<td>6</td>
<td>111</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>AAGACATTGATCGGGCTGG</td>
<td>115</td>
<td>10</td>
<td>120</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>ACACCAAGTTCCAGCTGACA</td>
<td>116</td>
<td>8</td>
<td>118</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>ACACCGTCCGATCTGGATAC</td>
<td>113</td>
<td>10</td>
<td>105</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>ACAGGCGAAGCGAAAATAGA</td>
<td>109</td>
<td>8</td>
<td>115</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>ACTTTGAGTCCTCATCCCGT</td>
<td>122</td>
<td>8</td>
<td>111</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>ATTGTGGCTAAGGAGCTCT</td>
<td>122</td>
<td>8</td>
<td>113</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>CAACCGGACAGTATGAGCAT</td>
<td>110</td>
<td>12</td>
<td>115</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>CGCGGATGTAGTGGCTTAC</td>
<td>115</td>
<td>10</td>
<td>123</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>GGCTTGAAGAGATGTCTCT</td>
<td>112</td>
<td>8</td>
<td>112</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>GTGATGCTGGATGAGGCGG</td>
<td>111</td>
<td>10</td>
<td>111</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>TTGCTACAGCGGAAAGGG</td>
<td>115</td>
<td>8</td>
<td>116</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.5: DNA sequences from BCA [31]
A set of 14 DNA sequences with \( L=20 \) have been reported in [34]. Table 5.6 shows the DNA sequences from [34] and their corresponding evaluation function values.

<table>
<thead>
<tr>
<th>DNA Sequence</th>
<th>H-measure</th>
<th>Self H-measure</th>
<th>Similarity</th>
<th>GC content</th>
<th>Continuity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTGACTTGAGGTAGGTAGGA</td>
<td>113</td>
<td>2</td>
<td>111</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>ATCATACTCCGGAGACTACC</td>
<td>115</td>
<td>6</td>
<td>116</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>CACGTCCTACTACCTTCAC</td>
<td>114</td>
<td>10</td>
<td>113</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>ACACGCGTGCATATAGGCAA</td>
<td>121</td>
<td>8</td>
<td>112</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>AAGTCTGCACCGATTCCTGA</td>
<td>115</td>
<td>10</td>
<td>113</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>AAGCCGAAGTTGACGTAAGA</td>
<td>119</td>
<td>8</td>
<td>111</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>CGACACTTGAGACACACCTT</td>
<td>120</td>
<td>8</td>
<td>125</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>TGGGCGCTCTACCGTGAATT</td>
<td>114</td>
<td>8</td>
<td>116</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.6: DNA sequences from GA [34]

A set of 7 DNA sequences with \( L=20 \) have been reported in [32]. Table 5.7 shows the DNA sequences from [32] and their corresponding evaluation function values.

<table>
<thead>
<tr>
<th>DNA Sequence</th>
<th>H-measure</th>
<th>Self H-measure</th>
<th>Similarity</th>
<th>GC content</th>
<th>Continuity</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGCTGAGACGCTGTATCGA</td>
<td>60</td>
<td>8</td>
<td>53</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>TATCCTAGCATGCTGCC</td>
<td>58</td>
<td>10</td>
<td>56</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>CAGTCTTGCTATCACATAC</td>
<td>54</td>
<td>10</td>
<td>53</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>CAGCGTGTCATATCGTGT</td>
<td>57</td>
<td>10</td>
<td>50</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>GATGCAGACGCGTATAC</td>
<td>53</td>
<td>12</td>
<td>49</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>GCTATAGTGACGACAGTAC</td>
<td>59</td>
<td>8</td>
<td>55</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>ATAGAGATCTACGCTGGCT</td>
<td>57</td>
<td>12</td>
<td>52</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.7: DNA sequences from EA [32]
It can be observed that sequences from BCA [31] and GA [34] and EA [32] have no continuity in the sequences which is desired and also the GC content value is equal to L/2. However, in all the papers, the DNA sequences show a higher H-measure, self H-measure and similarity values, which indicates that the possibility of false hybridizations will be higher compared to those generated in this work using SA approach. This comparison is pictorially represented in Figures 5.10 and 5.11. Average values of modified H-measure, self-measure and similarity are taken for this comparison.

![Figure 5.10: Comparison of SA with BCA [31] and GA [34]](image-url)
Figure 5.11: Comparison of SA with EA [32]
Chapter 6
Simulation of DNA cryptography algorithms

6.1 Overview

The DNA cryptography algorithms discussed in the previous chapters have been simulated using an object oriented approach. An object-oriented approach to simulating a system allows the system to be divided into sub-systems that can be simulated independently [36].

In the context of DNA computing, we can consider a test tube to be a system, since all the operations happen in the test tube. The subsystems can be considered as DNA sequences. Hence an object oriented approach was selected for simulating the cryptography algorithms that are based on DNA computing. In an object oriented approach, an object encapsulates all the simulation details, allowing other objects to exchange information only through an interface, without letting them know the implementation details. This is a very useful feature that can be used for simulating the DNA computing system to model the DNA sequences and their operations.

6.1.1 Implementation details

The simulation has been carried out using Java. The specific details of hardware and software configurations are given below.
**Hardware configuration**

The computer on which the simulation has been run had the following hardware configurations:

- 2.3 GHz Intel i3 processor
- 4GB of RAM
- Windows 7 operating system

**Software used**

The following software has been used for implementation of the java code.

- Java SDK 7
- Netbeans IDE

6.2 Classes used and their relationships

In object oriented approach, a class defines its data and function members which allow the objects, which are instances of class, to have a state and behavior. The development of various classes used in this simulation and their structure is discussed in this section. A class diagram defining the relationships between the classes is also discussed.

6.2.1 Description of the classes used

A brief description of the classes used and their attributes are discussed in this section.
**DNABase**

The DNABases class represents the valid DNA bases i.e., Adenine (A), Guanine (G), Thyamine (T) and Cystosine (C). Each base is represented as a single character. The two methods in the class are to check if a particular base in the DNA sequence is valid or not and to get the complement of the base. The class is represented pictorially in Figure 6.1

![Figure 6.1: Representation of DNABase class](image)

**ssDNA**

The ssDNA class represents the DNA single strand molecule. For the purpose of DNA computations, a single strand of DNA molecule is considered as a string. The main attributes of this class are the DNA sequence, directionality of the sequence, length of the sequence and GC content of the sequence. The operations in this class include checking if the sequence is valid, generate a random DNA sequence for a given length, GC content and direction, generate the complement and reverse complement of the sequence. This is represented in Figure 6.2
**dsDNA**

The dsDNA class represents the double stranded DNA molecule. The upper sequence in the double strand is considered as a string. It is assumed that several lower strands can attach to the string at different positions to the upper strand. Hence, an ArrayList data structure is used to contain the lower sequences and the position indices at which they attach to the upper string. One of the methods of dsDNA is to update the position at which a lower sequence gets attached to the upper sequence. The characteristics of the dsDNA class are shown in Figure 6.3

**ssDNAOperations**

All the single stranded DNA operations like annealing, separation based on length, ligation are implemented in this class.
Figure 6.4 shows the characteristics of this class.

![Figure 6.4: Representation of ssDNAOperations class](image)

**dsDNAOperations**

All the double stranded DNA operations like denaturation and PCR are implemented in this class. Figure 6.5 shows the characteristics of this class.

![Figure 6.5: Representation of dsDNAOperations class](image)

**generateDNAEncodings**

This class is used to implement the simulated annealing approach to generate reliable DNA encodings. The data members of this class are required population size and the length of the DNA sequences. Methods in this class are those required for implementing SA. Figure 6.6 shows the characteristics of this class.

![Figure 6.6: Representation of generateDNAEncodings class](image)
BinPCRCryptography

This class is used to encapsulate the data and members required to simulate the PCR based cryptographic algorithm. The data members would be plaintext and ciphertext and important methods of this class are for encryption and decryption of data. Figure 6.7 shows the characteristics of this class.

<table>
<thead>
<tr>
<th>BinPCRCryptography</th>
</tr>
</thead>
<tbody>
<tr>
<td>- plainText : String</td>
</tr>
<tr>
<td>- cipherText : String</td>
</tr>
<tr>
<td>+ Encryption()</td>
</tr>
<tr>
<td>+ Decryption()</td>
</tr>
<tr>
<td>- keyGenerate()</td>
</tr>
<tr>
<td>- DNAEncode()</td>
</tr>
<tr>
<td>- DNADecode()</td>
</tr>
</tbody>
</table>

Figure 6.7: Representation of BinPCRCryptography class

xorCryptography

This class is used to encapsulate the data and members required to simulate the cryptographic algorithm based on linear self assembly process that has been proposed in this thesis. The data members would be plaintext and ciphertext and important methods of this class are for encryption and decryption of data. Figure 6.8 shows the characteristics of this class.

<table>
<thead>
<tr>
<th>xorCryptography</th>
</tr>
</thead>
<tbody>
<tr>
<td>- popSize : int</td>
</tr>
<tr>
<td>- seqLength : int</td>
</tr>
<tr>
<td>+ Encryption()</td>
</tr>
<tr>
<td>+ Decryption()</td>
</tr>
<tr>
<td>- generateEncodings()</td>
</tr>
<tr>
<td>- keyGenerate()</td>
</tr>
<tr>
<td>- xor()</td>
</tr>
</tbody>
</table>

Figure 6.8: Representation of xorCryptography class
6.2.2 Class diagram

A brief analysis of the relationships between all the classes used for the DNA cryptography simulation is presented here.

To start with, a single stranded DNA molecule is composed of the four DNA bases A, G, T and C and is represented as a string. Hence ssDNA class and the DNABase class have a composition relationship. A double stranded DNA molecule is composed of ssDNA molecules indicating a composition relationship between them. When two Watson-Crick complementary ssDNA molecules come together, they form a double stranded DNA molecule.

The ssDNA operations class implements the operations possible on ssDNA and hence those two classes are related through a dependency relationship. In a similar way, dsDNA operations class is a dependency of dsDNA class. There is an association between the generateDNAencodings class and ssDNA class, since the former class generates a set of DNA single strands for computation.

The BinPCRCryptography is associated with the dsDNAOperations class since it uses PCR for the encryption process. The xorCryptography class is associated with the generateDNAencodings class, ssDNAOperations class and dsDNAOpertaions class.

The class diagram representing the relationships between all the classes are shown in Figure 6.9.
Figure 6.9: Class diagram showing the relationship between different classes
6.3 DNA operations

6.3.1 Implementation details

The DNA operations that have been implemented for this simulation are annealing, denaturation, PCR, separation based on L and ligation. The implementations are briefly described here. In the following discussion, a DNA single strand (ssDNA) is denoted as a DNA sequence for convenience.

**Annealing**

The annealing function takes an upper DNA sequence and a set of lower DNA sequences such that they are complementary to the upper sequence as input. Each lower sequence is then searched for in the upper sequence and the index of the match is obtained. The output is a dsDNA object that specifies the upper sequence and the ArrayList of lower sequences and their position indices. Time complexity of annealing is $O(n)$, where $n$ is the number of lower sequences to be annealed.

**Denaturation**

This function takes a double stranded DNA sequence as an input and cleaves the DNA molecule. It essentially returns an ArrayList containing the upper sequence and all the lower sequences. The time complexity of denaturation is $O(n)$, where $n$ is the number of lower DNA sequences.

**lengthSeparate**

This function takes a set of DNA sequences and returns the sequence that has the shortest length or longest length as specified. The time complexity is $O(n\log n)$ since
the length of each sequence needs to be sorted out using a sorting algorithm to find the sequence of required length.

**PCR**

PCR operation takes place in three steps namely denaturation, annealing and extension. As seen above denaturation and annealing take \( O(n) \) time steps. The extension steps takes \( L \) steps, where \( L \) is the length of the DNA sequence that needs to be extended. In PCR, for denaturation \( n=1 \) and for annealing, \( n = 2 \) since only 2 primer molecules need to be annealed, they can be considered as constant time operations. Hence the time complexity can be taken as \( O(L) \).

**Ligation**

The ligation function takes a set of DNA sequences and linker sequences as input and returns the upper sequence of the double stranded DNA that is formed as a result of ligation. This is done by iterating over the linker molecules and finding out the two DNA sequences that they link. LinkedList data structure is used to link the DNA sequences accordingly. The time complexity of the ligation operation is \( O(n^2) \) since for every linker molecule, it has to search all the DNA sequences to find the two sequences that it would link.

**6.3.2 Comparing time complexities**

In experimental implementation of DNA computing, the operations happen in a massively parallel manner and hence each operation represents a single lab step i.e., time complexity for each operation is \( O(1) \) in terms of lab steps. Table 6.1 shows the time complexities of each operation in computer simulation, experimental implementation and an estimation of real time it requires.
<table>
<thead>
<tr>
<th>Operation</th>
<th>Time complexity in computer simulation</th>
<th>Time complexity in terms of lab steps</th>
<th>Real time required (in hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annealing</td>
<td>O(n)</td>
<td>O(1)</td>
<td>4 hrs</td>
</tr>
<tr>
<td>Denaturation</td>
<td>O(n)</td>
<td>O(1)</td>
<td>3 hrs</td>
</tr>
<tr>
<td>Length Separate</td>
<td>O(nlogn)</td>
<td>O(1)</td>
<td>2 hrs</td>
</tr>
<tr>
<td>PCR</td>
<td>O(L)</td>
<td>O(1)</td>
<td>2 hrs</td>
</tr>
<tr>
<td>Ligation</td>
<td>O(n^2)</td>
<td>O(1)</td>
<td>12 hrs</td>
</tr>
</tbody>
</table>

Table 6.1: Time complexities in computer simulation, number of lab steps and real time required for various DNA operations

6.4 DNA cryptography algorithms

PCR based cryptography algorithm and the XOR cryptography based on linear self assembly proposed in this thesis are simulated using the DNA operations described above. The following sections give the implementation details of both these algorithms.

6.4.1 PCR based cryptography

The size of the plain text that can be entered for encryption is limited to 128 characters due to hardware restrictions.

Encryption algorithm

```plaintext
function Encryption(plaintext, primer1, primer2)
Input: plain text message in ASCII characters and the two primers that act as keys
Output: Encrypted message as a DNA sequence
    binaryText ← ASCIIToBinary(plaintext)
    encodedSequence ← digitalDNAAEning(binaryText)
    encodedSequenceWithPrimer ← addPrimer(encodedSequence)
    cipherSequence ← addRandomDNA(encodedSequenceWithPrimer)
end Encryption
```
In the encryption algorithm, if \( n \) is the number of characters in the plain text message, then the binary text will be \( 8n \) long. Digital DNA encoding function takes \((8n)/2\) time steps since every 2 binary bits are encoded as a DNA base. Adding primer and Random DNA would be of constant time. Hence, the time complexity of the encryption algorithm is \( O(n) \), where \( n \) is the number of ASCII characters in the plaintext.

The output is generated as a text file. A sample output for the input plain text “Crypto” is shown in Figure 6.10. Since long random DNA sequences are flanked on both sides, it is not possible to show the entire sequence and hence only a part of it is shown.

---

**Decryption algorithm**

The decryption algorithm is given below:

```
function Decryption(cipherSequence, primer1, primer2)
Input: Encrypted message as a DNA sequence and the two primers that act as keys
```
Output: plain text message in ASCII characters

messageSequence ← PCR( cipherSequence, primer1, primer2)

binaryText ← digitalDNADecoding( messageSequence )

plainText ← binaryToASCII( binaryText )
end Decryption

In the decryption algorithm, if n is the number of characters in the plain text message, PCR function would take \((8*n/2)\) steps (as derived above). Digital DNA decoding function takes \((8*n)/2\) time steps since every base is decoded back as 2 binary bits. Hence the time complexity of the encryption algorithm is \(O(n)\), where \(n\) is the number of ASCII characters in the plaintext.

The output is generated as a text file. A sample output for the above input plain text “Crypto” is shown in Figure 6.11.

<table>
<thead>
<tr>
<th>Sequence Amplified by PCR:</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTCATAAGAGTCTGCTCAATAACTCAGTCGTTCAATCTATGCCAATCTGATCAGTAGCTAGACT</td>
</tr>
<tr>
<td>Decrypted text in Binary:</td>
</tr>
<tr>
<td>0100001101110011001111110010001101111</td>
</tr>
<tr>
<td>Decrypted text:</td>
</tr>
<tr>
<td>Crypto</td>
</tr>
</tbody>
</table>

**Figure 6.11: Sample output for PCR based decryption**

6.4.2 XOR cryptography based on linear self assembly of DNA

The size of the plain text that can be entered for encryption is limited to 128 characters due to hardware restrictions.
**One-time pad generation**

The algorithm for generating a random one-time pad, which acts as the key is given below:

```plaintext
function keyGeneration(plaintext)
Input: plain text message in ASCII characters
Output: Random one-time pad DNA sequence

plaintextLength ← length(plaintext)
DNAsequences[] ← generateDNAsequences(8*plaintextLength,20)
DNAencodings[], linkerEncodings[] ← generateEncodingSeq(DNAsequences[])
key ← ligate(DNAencodings[], linkerEncodings[])
end keyGeneration
```

In the above algorithm, the `generateDNAsequences` function generates a set of reliable DNA sequences that can be used for the cryptography process. This function can be run once initially and can be stored for repeated usage later on. A sample output of this function is shown in Figure 6.12.

![Codebook of DNA sequences](image)

**Figure 6.12:** Screenshot of sample codebook of DNA sequences generated using the `generateDNAsequences` function

`generateEncodingSeq` takes the set of DNA sequences to generate the encoding sequences required for the ligation process. This has been explained in detail in
chapter 4. The time complexity of the algorithm can be deduced to be $O(n^2)$ as it mainly depends on the ligation operation. Screenshot of the output file for the `generateEncodingSeq` function for the input message “Crypto” is shown in Figure 6.13.

![Screenshot of sample DNA sequences generated using the `generateEncodingSeq` function](image)

Figure 6.13: Screenshot of sample DNA sequences generated using the `generateEncodingSeq` function
Screenshot of the key generated using the keyGeneration function is shown in Figure 6.14.

```
Key:
CGTCTGATCTGATGACTGACTGAGAGACAGTGTAGCAGCTGAGCATGAGATGAGTGACTGACGGCCTAGATGAGTGTAGTGCATCGTCCGGG
```

**Figure 6.14: Screenshot of sample output of the KeyGeneration function**

**Encryption algorithm**

```python
function Encryption(plaintext, key)
Input: plain text message in ASCII characters and the one-time pad that acts as key
Output: Encrypted message as a set of DNA sequences
    binaryText ← ASCIIToBinary(plaintext )
    encodedSequences[] ← DNAEncoding(binaryText )
    cipherSequences [] ← XOR(encodedSequences[], key)
end Encryption
```

The binary message will be encoded as DNA strings using the function `DNAEncoding`. The XOR operation essentially contains an annealing operation as discussed in chapter 4. Hence, the time complexity of this algorithm would be $O(n)$, where $n$ is the number of bits in the binary message. The resulting cipher sequences can be converted to a binary string if the correct encoding sequences are known. The cipher text constructed after encryption from the DNA sequences for the plain text “Crypto” is shown in Figure 6.13.

```
CipherText:
11100001110001000111000101100111001110011101000
```

**Figure 6.15: Cipher text constructed from the DNA sequences after XOR encryption**
**Decryption algorithm**

The decryption algorithm is given below:

```plaintext
function Decryption(cipherSequence, key)
Input: Cipher sequences and key
Output: plain text message in ASCII characters
    plainSequences ← XOR(cipherSequences, key)
    binaryText ← sequenceToBinary(plainSequences)
plainText ← binaryToASCII(binaryText)
end Decryption
```

The decryption algorithm is similar to that of the encryption. Hence the time complexity is $O(n)$, where $n$ is the number of bits in the binary message. The plain text constructed after decryption from the DNA sequences is shown in Figure 6.14.

![Decrypted Text in Binary: 010000110111001001111001011100000111010001101111 Decrypted Text: Crypto](image)

**Figure 6.16: Plain text constructed after the XOR decryption**

### 6.4.3 Results

Both the cryptography algorithms are compared with respect to the time complexity in simulation, number of lab steps required and the real time required. This is shown in Table 6.2.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time complexity in simulation</th>
<th>Number of lab steps required</th>
<th>Real time required in hours</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PCR based cryptography</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key generation</td>
<td>Constant time</td>
<td>Not required</td>
<td>-</td>
</tr>
<tr>
<td>Encryption</td>
<td>O(n)</td>
<td>1 (Synthesis)</td>
<td>Depends on the length of the molecule to be synthesized</td>
</tr>
<tr>
<td>Decryption</td>
<td>O(n)</td>
<td>2 (PCR, extraction)</td>
<td>~4hrs</td>
</tr>
<tr>
<td><strong>XOR cryptography</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key generation</td>
<td>O(n)</td>
<td>2 (Synthesis, ligation)</td>
<td>&gt; 12hrs</td>
</tr>
<tr>
<td>Encryption</td>
<td>O(n)</td>
<td>3 (Annealing, extraction, synthesis)</td>
<td>&gt; 4hrs</td>
</tr>
<tr>
<td>Decryption</td>
<td>O(n)</td>
<td>3 (Annealing, extraction, synthesis)</td>
<td>&gt; 4hrs</td>
</tr>
</tbody>
</table>

**Table 6.2: Comparison results for PCR based and linear self assembly based cryptography algorithms**

**Discussion**

Though PCR based cryptography requires fewer lab processing steps and time, it may be prone to high error rates due to the fact that a random DNA sequence would be flanked on both sides of the message. If the primers are not chosen properly, this could lead to high probability of occurring of false hybridizations during the PCR process which might eventually give a wrong cipher text. The false hybridization errors in XOR cryptography proposed in this thesis can be avoided by selecting the sequences carefully and hence, the error rates in this process are relatively low.
Chapter 7
Conclusions and future work

7.1 Conclusions

In this thesis, several DNA cryptographic algorithms were studied and analyzed to understand the reliability of the encryption and decryption processes. DNA computing techniques are, in general, prone to errors due to the use of biological reactions. Hence, for a DNA encryption algorithm, it is important to assess its reliability along with the security it provides. All the algorithms studied and described in this thesis exhibit a highly secure way of encrypting data, but may be subject to several hybridization errors.

A new method to encrypt data based on linear self-assembly of DNA has been proposed in this thesis. Since, this method involves mostly hybridization reactions, the hybridization errors are greatly reduced by selecting reliable DNA encodings for encoding the binary message as well as generating the one-time pad sequences.

In this work, DNA encoding problem of generating reliable DNA sequences to prevent false hybridizations has been studied using the simulated annealing approach. The evaluation function defined takes care of minimizing all the mishybridization errors. Also, constraints to provide stable DNA molecules in terms of continuity and GC-content were defined. The SA algorithm used in this thesis has produced better results compared to that of the previous approaches to the problem for the combinatorial constraints defined in this thesis.
Finally, an object oriented simulation of two DNA cryptographic algorithms has been performed using the Java programming language. The simulation models the DNA operations at an abstract level. However, these functions have been defined in a flexible manner, such that they can be modified easily at a later stage. The cryptography algorithms that have been implemented were the PCR based cryptography [20] and the DNA self-assembly based cryptography defined in this thesis. The DNA self-assembly based cryptography takes more lab steps when compared to the PCR based approach. However, the false hybridization errors in self-assembly based cryptography proposed in this thesis can be avoided by selecting the sequences carefully and hence the error rates in this process can be considered to be relatively low compared to the PCR based cryptography.

7.2 Future work

For the DNA encoding problem, only combinatorial constraints have been considered in this thesis work. Better DNA sequences can be obtained by considering the thermodynamic constraints too.

The simulation model presented in this work defines a basic framework for using object-oriented approach to simulate DNA computing operations and algorithms that use these operations. The DNA operations in this work are defined at a higher level of abstraction and not at the molecular level. This simulation model can be made more robust by including the errors inherent in the DNA operations. By doing so, a more clear idea about the reliability of any DNA algorithm can be obtained. Also, the model can be made to simulate the DNA operations at a molecular level by adding several physical and biological properties from the literature. This would result in an accurate simulation model which will mimic the processes occurring in an actual test tube.
References


