DNA Cryptology: A Brief Survey And A DNA Algorithm For Breaking A Propositional Logic Based Cryptosystem

A dissertation submitted in partial fulfillment
of the requirements of M.Tech.(Computer Science)
degree of Indian Statistical Institute, Kolkata
by

Bireswar Das

under the supervision of

Professor Rana Barua
StatMath Unit

Indian Statistical Institute
Kolkata-700 108.

18th July 2003

Indian Statistical Institute

203, Barrackpore Trunk Road,

Kolkata-700 108.

Certificate of Approval

This is to certify that this thesis titled **"DNA Cryptology: A Brief Survey And A DNA Algorithm For Breaking A Propositional Logic Based Cryptosystem"** submitted by **Bireswar Das** towards partial fulfillment of requirements for the degree of M.Tech. in Computer Science at Indian Statistical Institute, Kolkata embodies the work done under my supervision.

Rana Barua

Professor Rana Barua
StatMath Unit
Indian Statistical Institute
Kolkata-700 108.

Acknowledgments

I take pleasure in thanking Dr. Rana Barua for his guidance, advice, encouragement and highly helpful criticism throughout the dissertation period. His pleasant and encouraging words have always kept my spirits up.

I would also like to thank Dr. Rani Siromoney for motivating me into this field and for granting me permission to include the paper [24] in this thesis.

My felicitation also goes to Shantanu Das for providing me all the help during the period I worked.

Bireswar Das
Indian Statistical Institute.
Kolkata-700 108

Abstract

The aim of the project is study the methods in DNA cryptology and to present an attack on a cryptosystem based on propositional logic which is presented in [10] and shown to be optimal in the sense that any cryptanalytic attack against it can be used to break other cryptosystems as well. Such a cryptanalytic method could also be used to solve in polynomial time, the membership problem of any language in NP∩Co-NP.

Tom Head [6] has given an elegant and simple DNA algorithm to solve the SAT and other hard problems. Artificial plasmids are created, which are circular double-stranded DNA molecules. The DNA computation is done by initializing with this single molecular variety combining the operations of cut and paste into a single compound operation.

Contents

Chapter 1

Introduction

The history of computing is not new; it started much before the advent of electronic computers. Though electronic computers have revolutionized the field of computing, it also has its limitation dictated by the laws of nature. At some point in future this limits will be reached. To break down this barrier, one possible way is to go to molecular level. In this respect Adleman was the first to show that DNA molecules could be used to perform computation.

Hardly eight years have gone by since Adleman introduced [1] the revolutionary idea of making use of DNA computations to solve difficult combinatorial problems. Lipton adopted a similar method and showed that SAT as well as other NP-hard problems could be solved using molecular computations [14]. These opened a new way of solving many problems in different fields, including cryptology. Efficient design of steganographic system, one-time-pad cryptosystem has be discussed in [5]. The problem of attacking the widely used Data Encryption Standard (DES) is addressed in [2, 4] and DNA computing is shown to have wide application in cryptography [3]. Application of the sticker model for the cryptanalytic attack on the DES is discussed and described in detail in [16]. Attacking NTRU with the help of DNA computing showed that not only brute force attack but other efficient attack can also be implemented in this paradigm.

In chapter 2 we discuss the structure of DNA molecule and describe the various operations that make DNA computing so powerful. Different subsets of these operations give rise to various models of DNA computation which has their own merits and demerits. We describe some of the models which are important for DNA cryptology in chapter 3. Chapter 4 is a survey of the existing methods in DNA cryptology. In chapter 5 we describe the DNA algorithm for breaking a propositional logic based cryptosystem [24].

Chapter 2

DNA Computing: An Introduction

2.1 Structure of DNA Molecule

DNA(deoxyribonucleic acid) is the master molecule in whose structure is encoded all the information needed to create and direct the chemical machinery of life. Though the existence of DNA molecule was confirmed much before, it was not until April 25, 1953 that the exact three dimensional double helix structure of DNA was known. This fine job was done by Watson and Crick. DNA is a polymer which is strung together from monomers called deoxyribonucleotide or simply nucleotide. Each nucleotide consists of three distinct chemical subunits: a five-carbon sugar, acidic phosphate and a nitrogen-rich base(B). The five carbones are named 1' to 5' as in Figure 2.1.

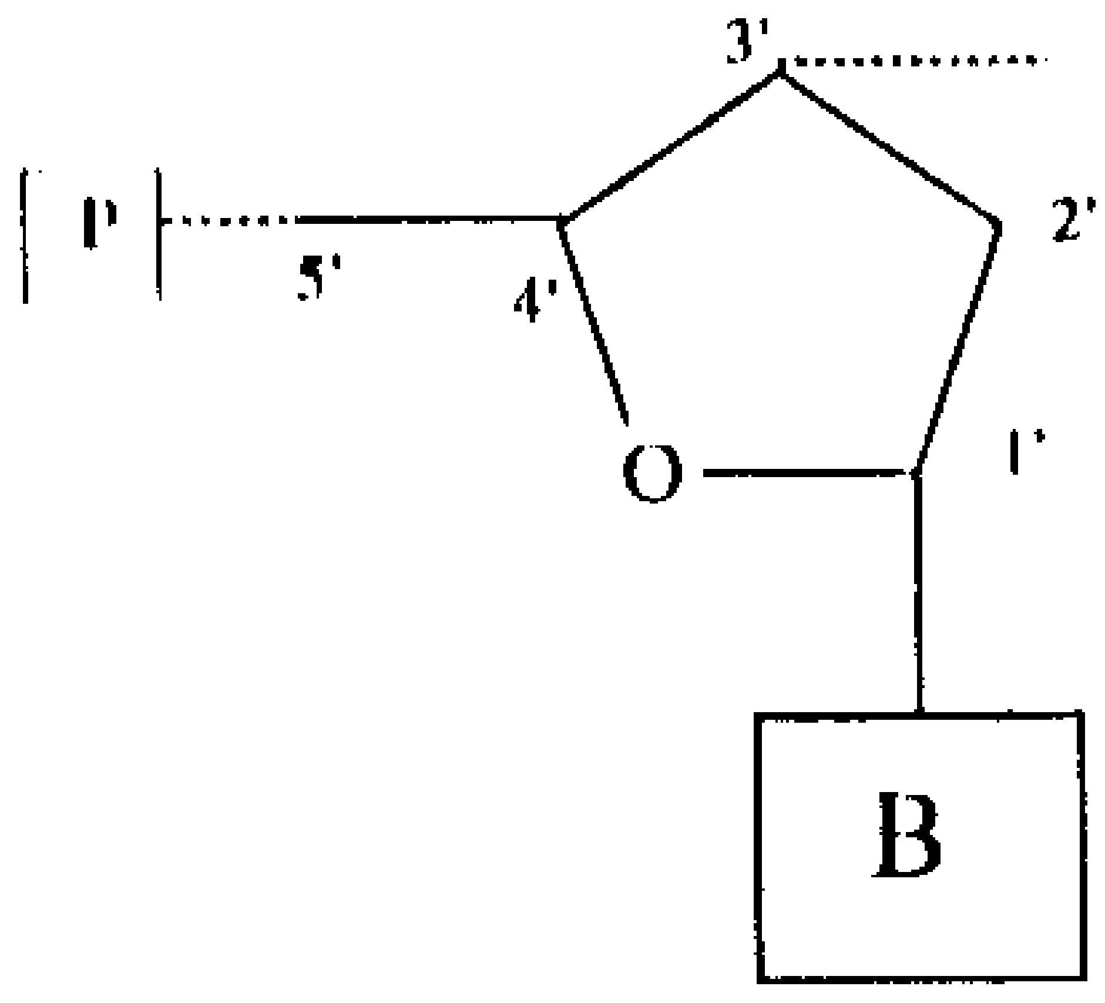

Figure 2.1: Schematic diagram of Nucleotide.

There are four types of bases: adenine, thymine, guanine, cytosine. Based on which base is present, a nucleotide is referred to as A, T, G, C.

Nucleotides can link in two different ways:

1) Through a strong covalent phosphodiester bond which joins a 5'-phosphate group of one nucleotide to a 3'-hydroxyl group of another nucleotide Figure 2.2.

2) Through a weak hydrogen bond between two base from two nucleotides Figure 2.3. *But* adenine(A) can pair with only thymine(T) and cytosine(C) can pair with only guanine(G). This paring principle is called *Watson-Crick complementarity* or WC complementarity.

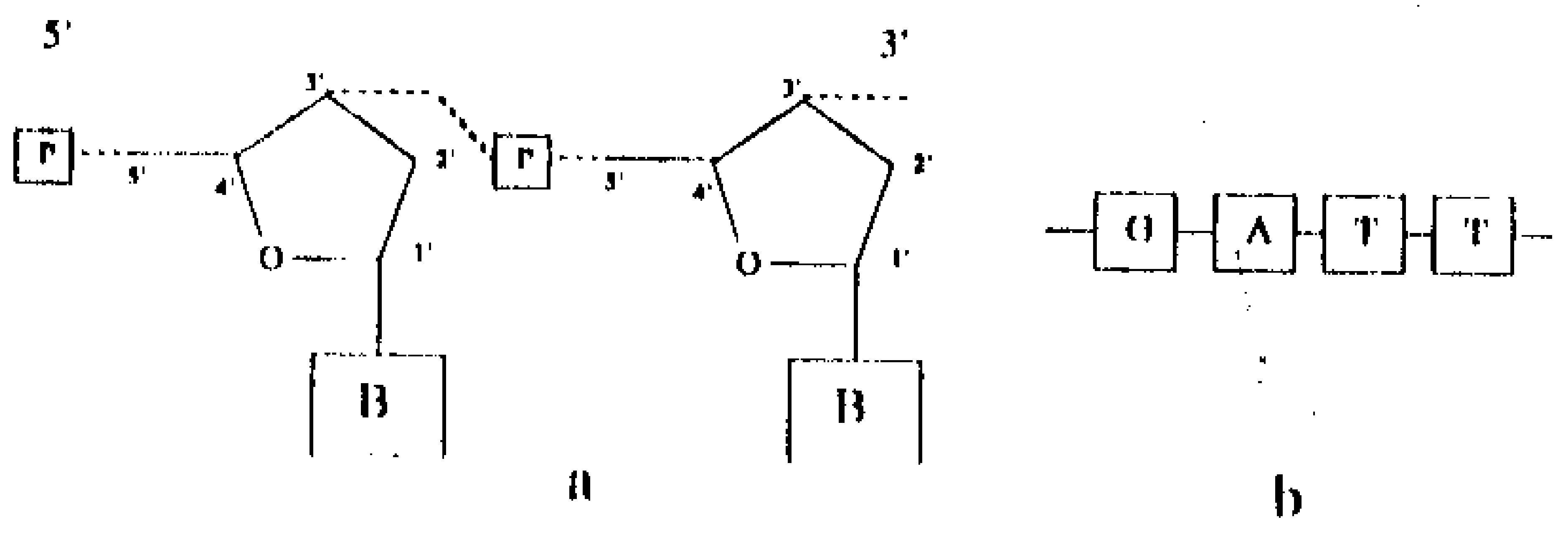

Figure 2.2: a) A phosphodiester bond. b) A schematic diagram of single strand DNA molecule.

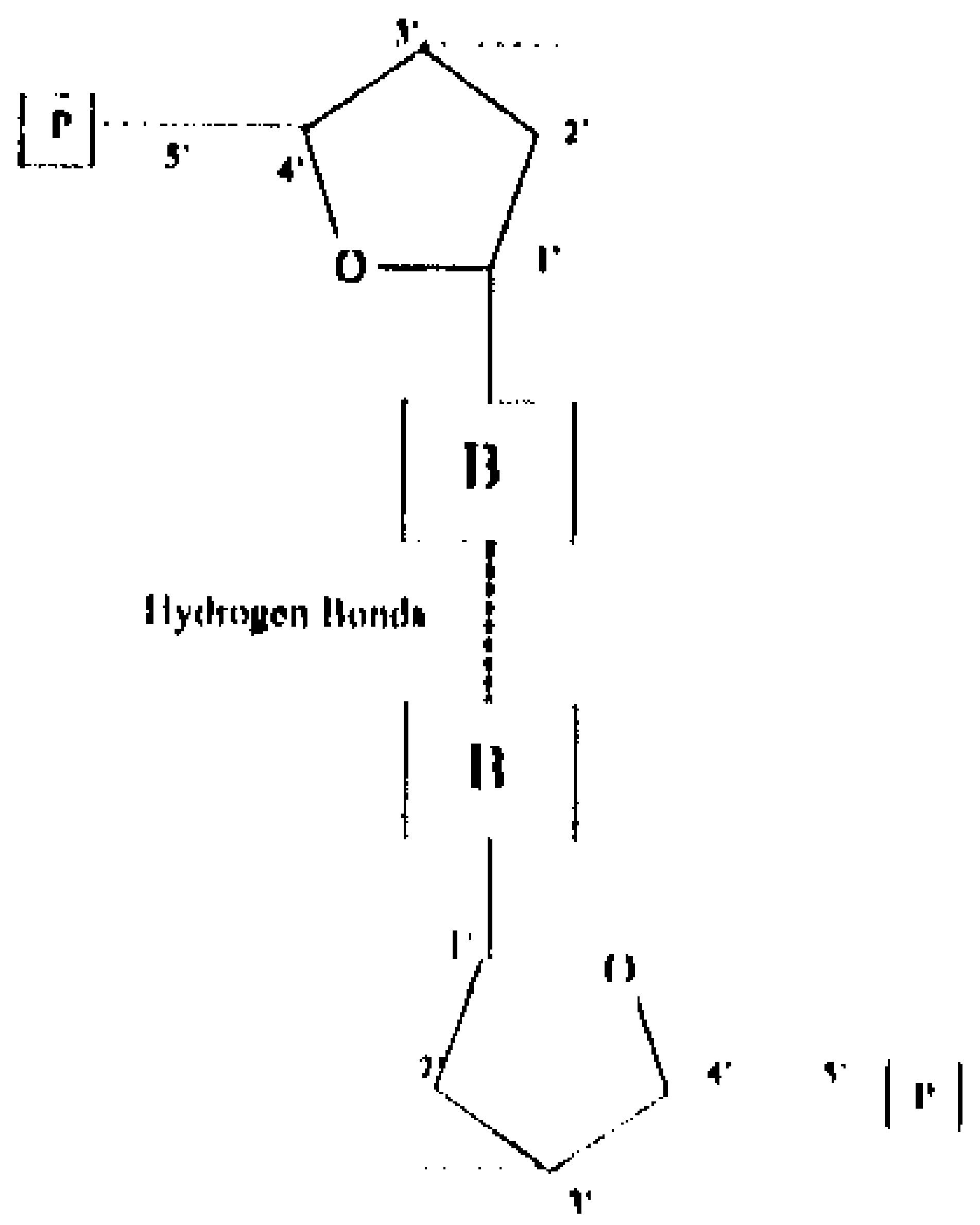

Figure 2.3: Hydrogen bonds between two bases.

Using only phosphodiester bond we get single strand DNA molecule Figure 2.2 b. The 5'-phosphate group and 3'-hydroxyl group induces an orientation in the DNA molecule Figure 2.2 a, which is called a 5'-3' molecule if we start from the 5'-phosphate group and go towards 3'-hydroxyl group or a 3'-5' molecule if we start from 3'-hydroxyl and go towards 5'-phosphate group.

If two single strands satisfy WC complementarity and they are opposite in orientation(i.e., one in 5'-3' orientation and the other in 3'-5' orientation) then the two strand can give a double strand DNA molecule Figure 2.4. But this molecule is not linear in structure; the two single strands are wound around each other to form the famous double helix.

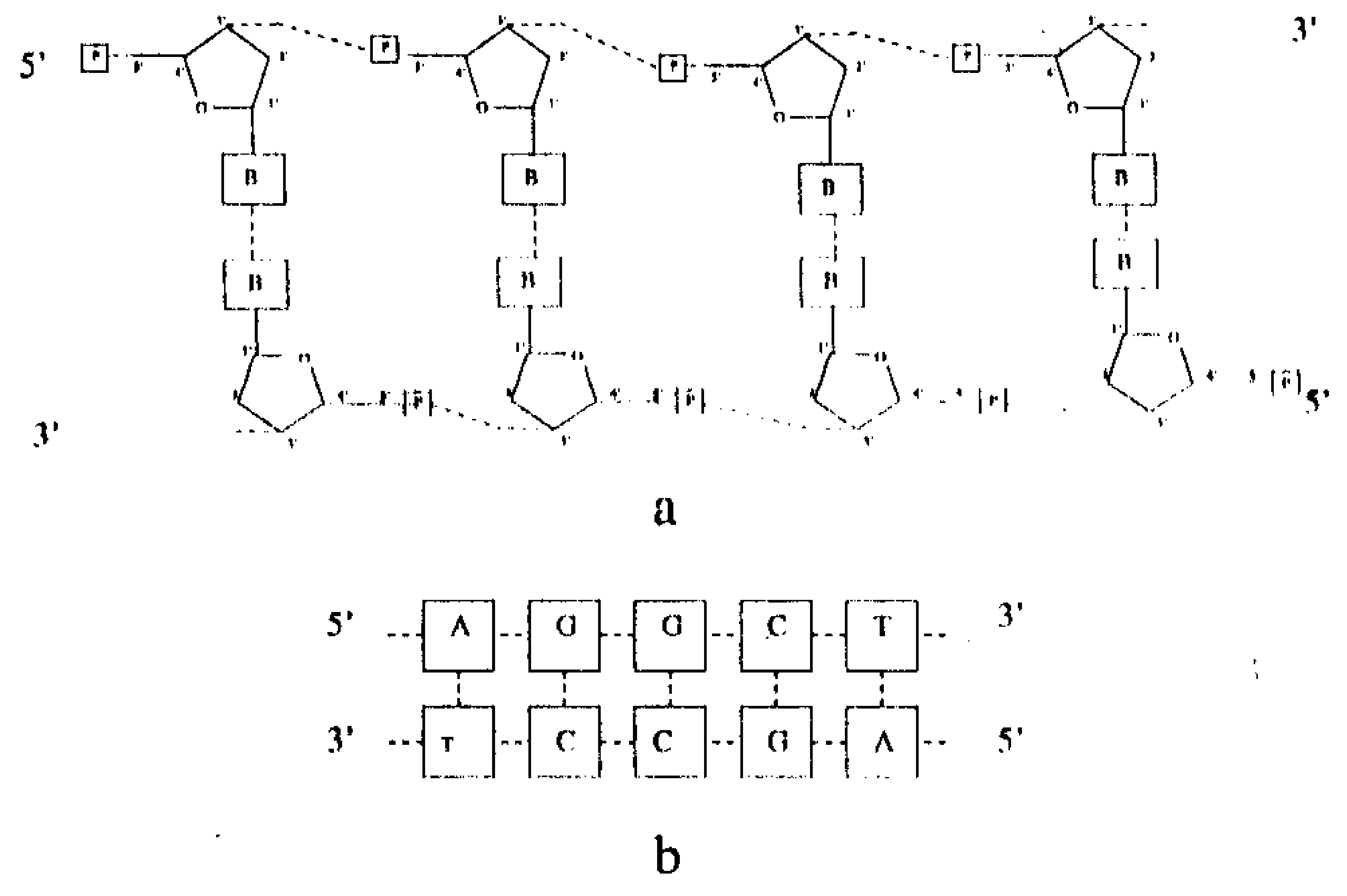

Figure 2.4: a) Double strand DNA molecule. b) Schematic Diagram.

2.2 Molecular Operations

DNA molecule can be manipulated in various ways. The different chemical operations serves as the tool kit for DNA computation.

2.2.1 Melt and Anneal

A double strand DNA molecule can be separated into two single strands by varying the temperature and pH of the solution. In this process the hydrogen bond between the bases breaks. This is called *melting*. The reverse process is called *annealing*. In annealing two single strands which satisfy WC complementarity and they are opposite in orientation could be combined to form a double strand DNA molecule. This is also affected by varying the temperature and pH of the solution which reestablishes the hydrogen bonds between the bases.

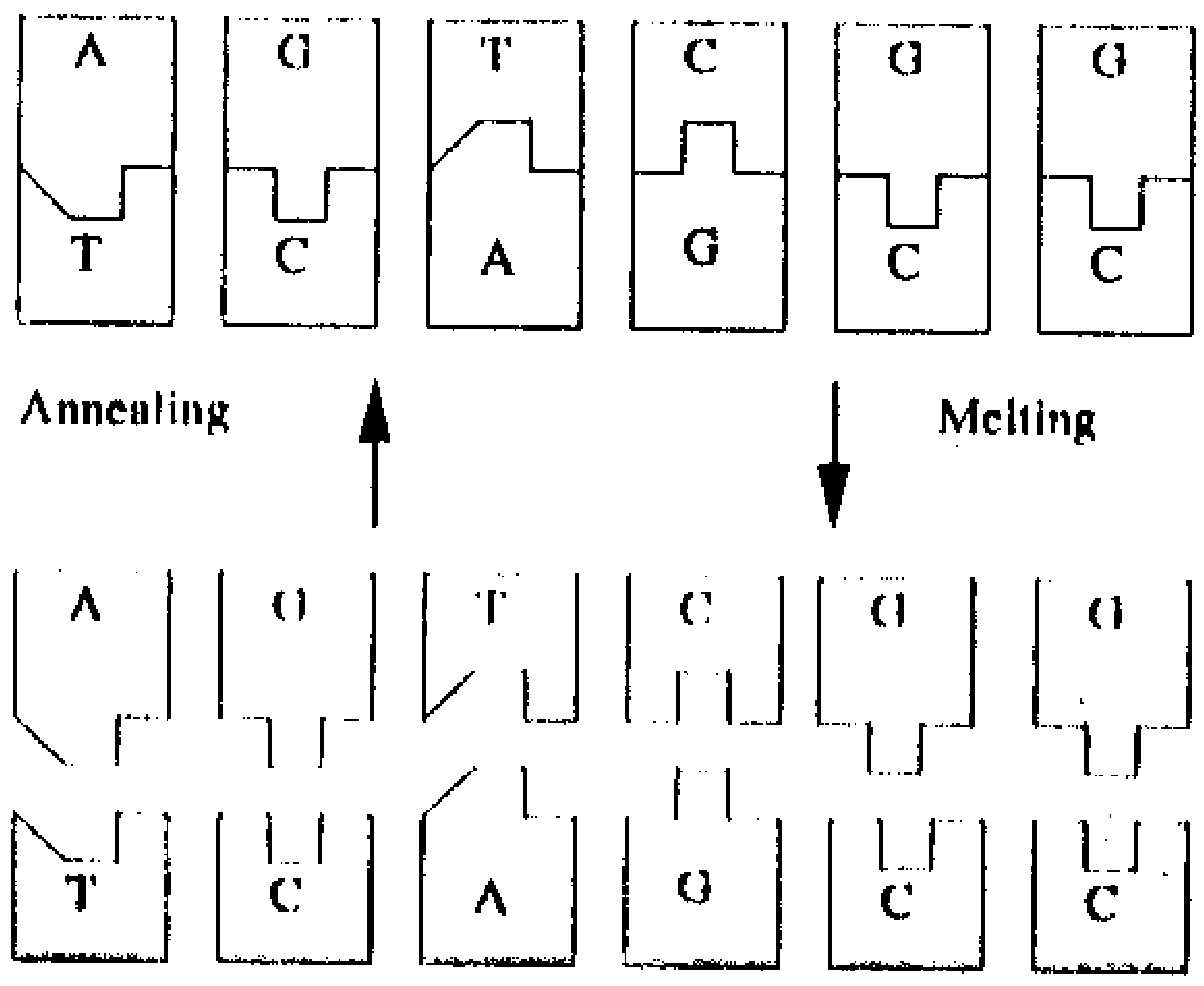

Figure 2.5: Annealing and Melting

2.2.2 Polymerize

We can form the WC complement of a single strand DNA molecule using an enzyme called DNA polymerase. The strand whose complement is to be made is called template. For this purpose, a short sequence of nucleotide, called the *primer* is attached at one end of the template. The enzyme polymerase then extends the sequence by adding nucleotide one by one to form the total WC complementary sequence.

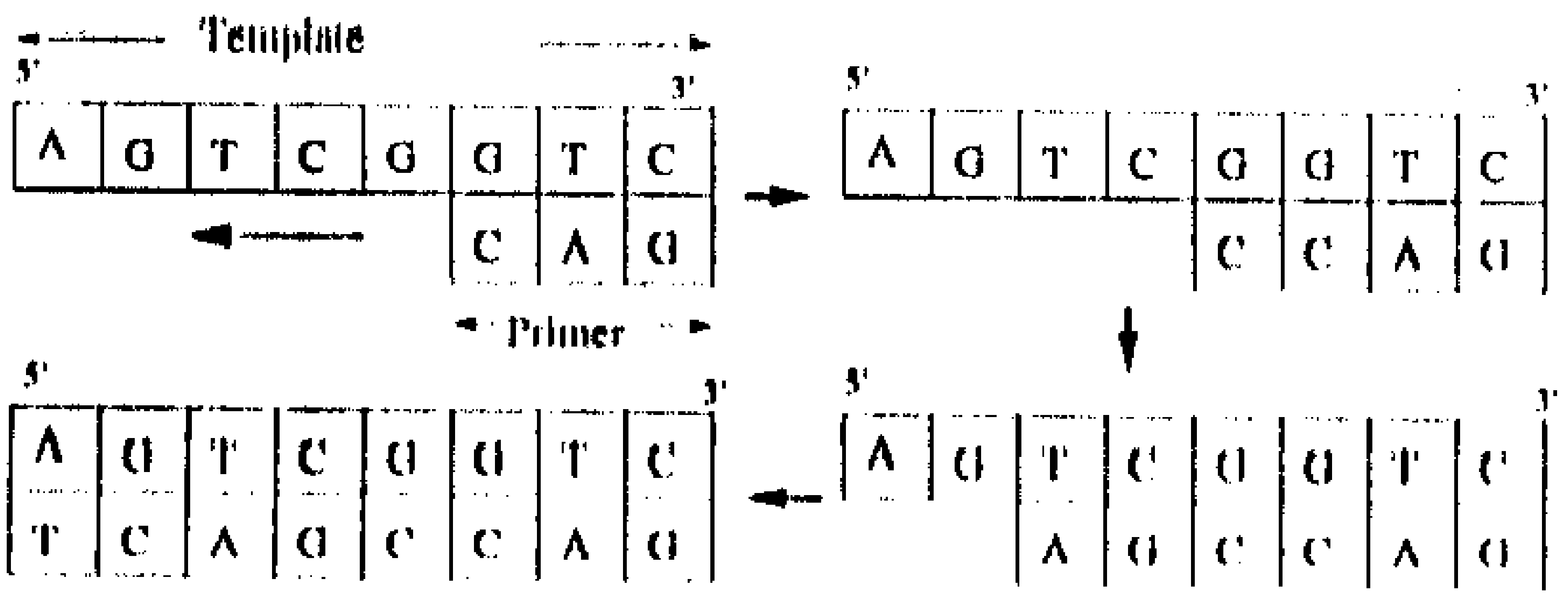

Figure 2.6: Polymerization : Extension occurs from 3' to 5' of the template.

2.2.3 Extract

In this process all strands containing a given substrand x can be extracted from a solution of different DNA molecules. This could be done using *biotin-avidin affinity purification* as in [1]. First copies of $\overline{x}$ (i.e., the WC complement of x), called the *probes*, are formed and they are attached to biotin molecules, which in turn will be anchored to an avidin bead matrix. If we then melt the strand in the solution and pour them over the matrix, strand containing x will be attached to the probes. The other strand are simply washed out.

2.2.4 Synthesis

Any short sequence of nucleotide can be synthesized in the lab. Longer random sequence are available in the DNA pool of various organisms.

2.2.5 Gel Electrophoresis

DNA molecules can be separated based on the lengths of the molecules. To do this some of the DNA solution, strained with *ethidium bromide*, is placed in a container filled with *agrose gel.* The container is then kept in an electric field. As the DNA molecules are negatively charged, they starts moving in the gel due to the electric field. The smaller molecules move a greater distance than the heavier molecules. Thus they form separate bands on the gel which can be viewed or can be separated to get the strands in the bands. With agrose gel strands differing in length one can also be separated [16].

2.2.6 Amplification Through PCR

With the technique called *polymerize chain reaction*(PCR) a strand can be replicated very efficiently and elegantly. Suppose we want to amplify a double strand molecule a with known borders b and c. First a solution containing a, the two primers b and c, polymerase enzyme and plenty of nucleotides is prepared. The strand a is then melted with the application of heat to form two complementary single strand a_1 and a_2 Figure 2.7. When the solution is cooled down the primers attaches at the ends. As in polymerization th primers then extends to form the whole strand due to the effect of polymerase. If we repeat the process we can amplify strand a exponentially.

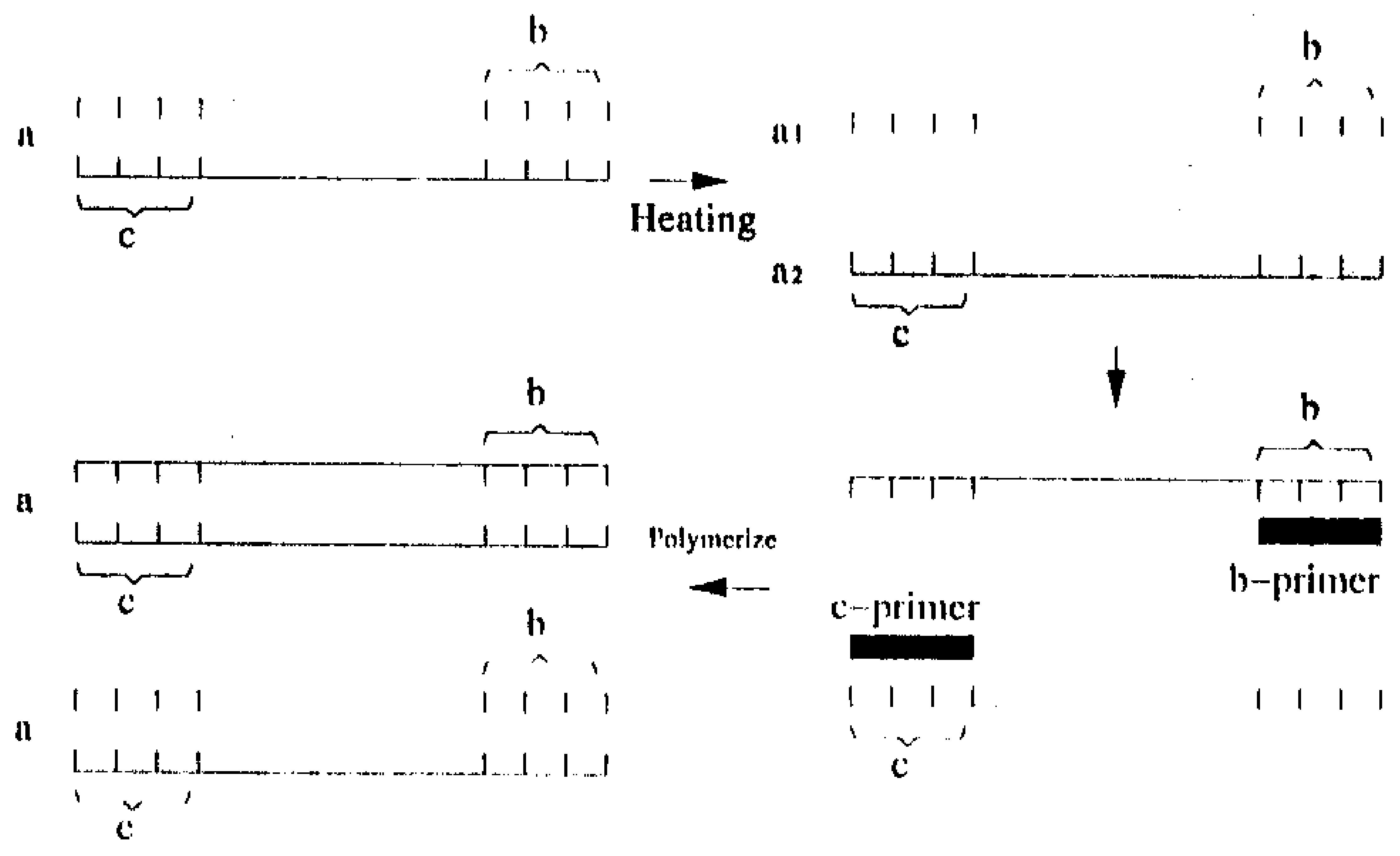

Figure 2.7: Steps of PCR

2.2.7 Detect

Given a test tube with a solution we can easily detect whether the test tube contains any strands or not using regents.

2.2.8 Cut and Ligate

There are enzymes that can cut DNA molecules in various ways. Endonuclease(SI endonuclease) can cut a double strand(resp. a single strand); but they are not site specific, i.e., the they can cut anywhere in the molecule. There is a class of enzymes called restriction enzymes which can cut double strand at specific sites having definite sequence of nucleotides. As an example we the action of EcoRI Figure 2.8. EcoRI acts on site having sequence GAATTC. This action is called *splicing*. Two separated strands can be with reunited by removing the restriction enzyme and adding enzyme called *ligase*. The general action of ligase is to create a phosphodiester bond. Two partial double strand with complementary overhanging sequence of nucleotide, called the *sticky end*, can be joined by applying ligase. This process is called *ligation*.

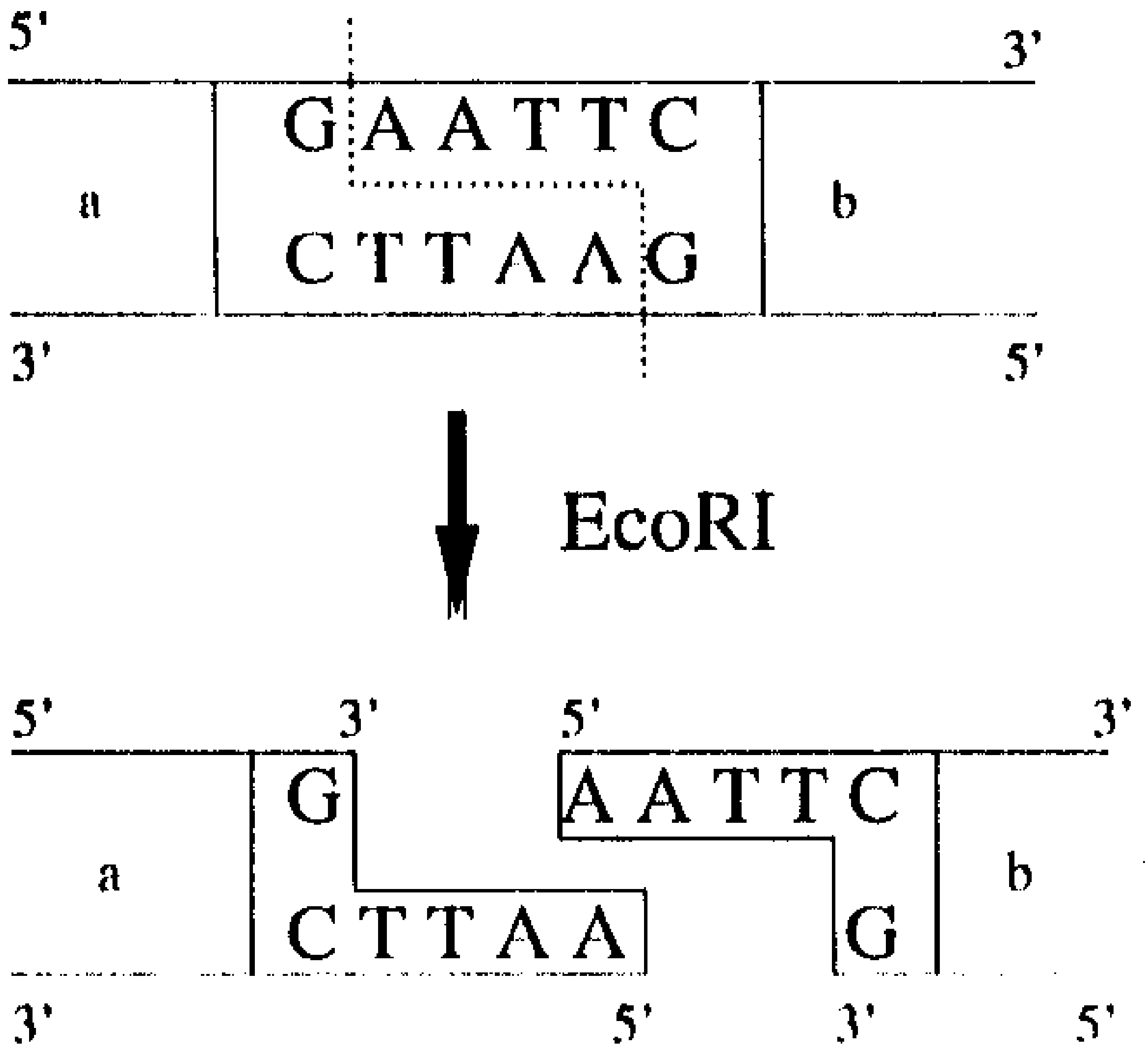

Figure 2.8: Splicing with EcoRI, a,b ∈ $\{A, G, T, C\}^*$

2.2.9 Sequencing

In this process an unknown DNA strand can be read to know the exact sequence of nucleotide. But sequencing is a costly affair and effort should be made to avoid this.

2.3 Some Points in Favour of DNA computing

1. DNA strand has the capability of processing information due to its chemical properties.

2. DNA strand can store an incredible amount of data in a very small volume.

3. It is massively parallel. This gives DNA computing very high speed.

4. Various complex structure like living organism is the result of applying few simple operation to an initial DNA sequence. This makes us believe that DNA can be a potential tool for computation.

2.4 Issues to be settled

DNA computing has some pitfalls which has to be settled before we can get a practical DNA computer. Errors in DNA computing may occur due to the fact that the intended DNA operation does not take place always. For example, one strand may anneal to some other strand in way making a projection like bubble. Likewise, extraction may not always be perfect. If the error rate is not kept within limit, the whole computation may be wrong or the volume required may be huge. As a demonstration, in Adleman's method of breaking DES if the error rate is 10^{-4}, the volume required will be only a few grams of DNA to perform the whole operation. In contrast if the error rate is 10^{-2} the volume required will be more than 23 Earth masses [16]. Although some results has been obtained to make DNA computation error resistant, lot of work is yet to be done.

Chapter 3

Models of DNA Computation

There are various models of DNA computation. Some of the them are mathematical models capturing the real world situation, some of which are purely mathematical whose experimental feasibility remains unresolved. Here we describe three models which has been used for the purpose of DNA cryptology.

3.1 Sticker Model

This model introduced in [22], was used [4] for breaking the widely used Data Encryption Standard. The model is based on WC complementarity. This model uses two kinds of strands referred to as *memory strands* and sticker strands or simply *stickers*. A memory strand is a n nucleotide long single strand consisting of k non-overlapping substrands (which are usually of equal length m). Each nucleotide of the memory strand may not be of a part of any of the k-substrands. Thus $n \geq mk$. Any two of the k-substrands differ in several base positions. A sticker is just the WC complement of exactly one of the k-substrands.

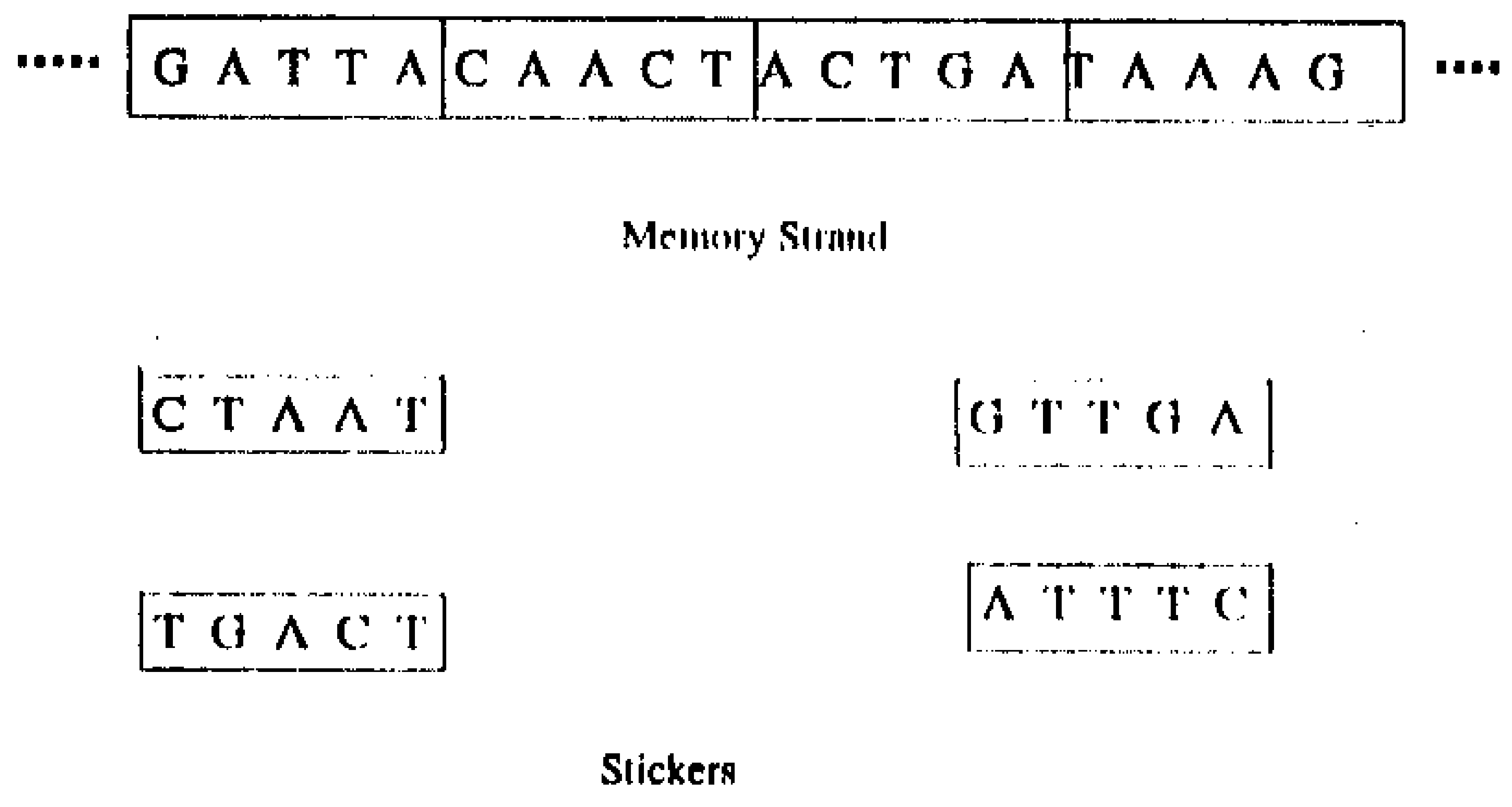

Figure 3.1: Sticker Model: Memory strand and Stickers.

If a sticker is attached to its matching substrand on the memory strand the particular substrand is said to *on*. Otherwise it is *off*. For $n = 20$, $k = 4$ and $m = 5$ we show how to represent 0000, 0101, 0110 in Figure 3.2.

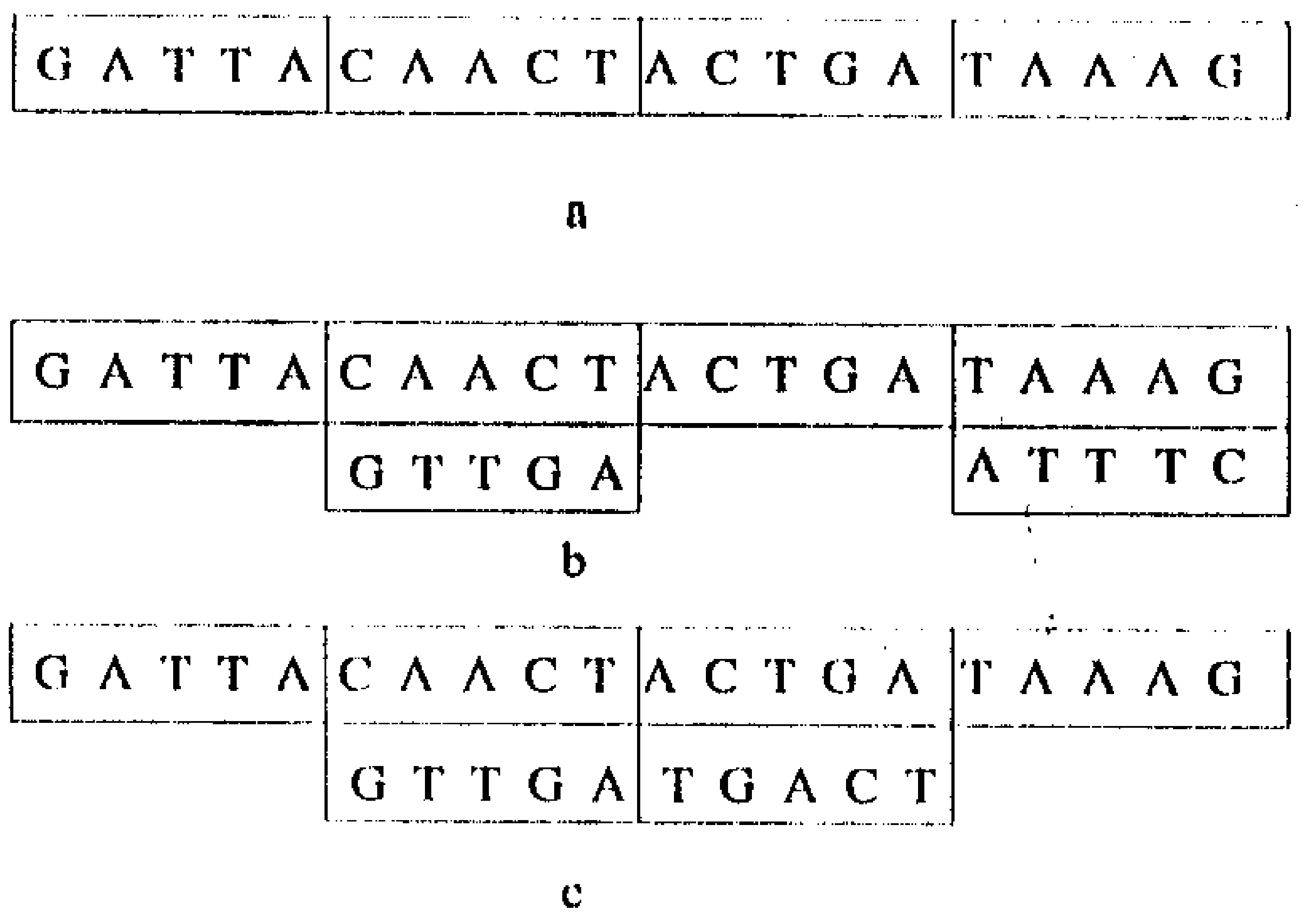
G A T T A C A A C T A C T G A T A A A G
a
G A T T A C A A C T A C T G A T A A A G
G T T G A
A T T T C
b
G A T T A C A A C T A C T G A T A A A G
G T T G A T G A C T
c

Figure 3.2: Memory complex representing a) 0000, b) 0101, c) 0110

3.1.1 Operation on Sticker model

Before specifying the operations on sticker model we define a *test tube* to be a multiset containing the memory complexes. The general operations on the memory complexes in a test tube are merge, separate, set and clear.

merge: Two test tubes are combined into one. This is just mixing the solution of two test tubes.

separate: Given a test tube T and an integer i, $1 \leq i \leq k$ this produces two test tube $+(T, i)$ and $-(T, i)$ where $+(T, i)$ $(-(T, i))$ contains all memory complexes whose i^{th} substrand is on (resp. off).

set: Given a test tube T and an integer i, $1 \leq i \leq k$ this produces another test tube $set(T, i)$ where each memory complex has its i^{th} substrand on.

clear: Given a test tube T and an integer i, $1 \leq i \leq k$ this produces another test tube $clear(T, i)$ where each memory complex has its i^{th} substrand off.

The input or initial test tube will be a library of memory complexes. In particular a (k, l) library, $1 \leq l \leq k$, consists of memory complexes with k substrands, the last $k - l$ substrands are on whereas the first l substrands are on and off in all possible ways. Thus, a (k, l) library contains 2^l different memory complexes.

3.2 Splicing System

Splicing system was proposed by Tom Head [8]. Splicing system captures mathematically the two molecular operation cut and ligate introduced in Chapter 2. The mathematical model was introduced and studied before Adleman's experiment. Many results including universality of splicing system was obtained in [7, 19, 20]. In several organisms the DNA present is circular. If both circular and linear DNA strands are present the mathematical analysis becomes much more complicated because then several different possibility must be handled. Despite of that various result concerning circular splicing has been obtained in [25, 21]. In Chapter 5 we describe a special type of circular splicing introduced in [ku:head], which has been used to solve various combinatorial problems. From practical point of view this splicing seems feasible because of easy availability of the particular type of circular DNA strands and the simple operations on it. This splicing has been used in [24] to break a public key cryptosystem.

3.3 Algorithmic Self-Assembly

Introduced in [29], self-assembly seems to be a powerful tool for DNA computing paradigm. Apart from creating the building blocks, self-assembly only involves annealing and ligation for computation. The building blocks are various complex nanoscopic structure of DNA molecules. Some of the nanoscopic structure has been studied and created by Seeman et al. Winfree showed that self assembly has the power of universal computation in [29].

In [15] it has been shown that nanoscopic structure using DNA could be made which in turn serves as the building block of self-assembly. This gives greater flexibility while retaining the advantage of massive parallelism inherent in DNA computing. This nanoscopic structure can act like *Wang tiles* [28]. The wang tiles are squares with colored edges. If the Wang tiles are allowed to cover the plane according to an additional rule that only edges of same color can face each other, then the Wang tiles can simulate Turing machine. Thus it has the power of universal computation. The DNA Wang tiles are "Double Crossover" or "Triple Crossover" tiles made up of several interwoven DNA strands to form a square body with sticky ends coming out of the sides or corners. This tile has been used to compute Xor in [12], multiplication and circular convolution in [18]. The problem of attacking the cryptosystem NTRU and creating DNA one-time-pad has been addressed in [18] and [5] respectively. The stability and error resistance of DNA tile seems promising for DNA computing.

Chapter 4

A Brief Survey on DNA Cryptology

In this chapter we discuss the application of DNA computing in cryptology. We show how the job of cryptography and cryptanalysis can immensely be helped using DNA computing.

4.1 Cryptography

4.1.1 DNA Steganography

Steganography is the method of hiding some message in some apparent innocent looking information. As an example, we can state the microdot method of steganography where the sender sends a message by first taking a photograph of the message and then reducing it to a size of a full stop. This greatly reduced photograph can be pasted in a letter in the place of a full stop and can be sent to the receiver. As DNA molecules can serve as an extremely compact storage medium(few grams of DNA can contain all the stored data in the world), DNA can be a potential tool for steganography and this has been shown first in [27] by Celland et. al.. In their method a DNA strand of the form p_1mp_2 is formed, where $p_1, m, p_2 \in \Sigma^+$ and $\Sigma = \{A, G, T, C\}$. p_1 and p_2 are the keys which also serves the purpose of primer in PCR in the decryption process. The message is encoded/encrypted in the DNA strand m. The strand p_1mp_2 is then mixed with several other DNA strands which are physically similar to p_1mp_2. The resulting collection of strands can be send as a microdot. The receiver just uses the primers(key) for PCR and amplifies the particular strand p_1mp_2. Then by gel electrophoresis the actual strand could be obtained. Even if the adversary knows the presence of the microdot then also he has to guess the primers to do the PCR. If the primers are of length 20 base pair then he has to try the PCR with more than 10^{20} primer pairs [27].

In [5] Gehani et. al. shows the limitation of the above steganographic method by an entropy based analysis. They also proposed some improvement over the system.

Another method of steganography was proposed in [13]. It is almost similar except that the primers are same for the redundant strands. The redundant strands as a collection (called the dummy pool), serves the purpose of the key. To decrypt, the receiver does PCR on the dummy pool(i.e., not containing the message strand) and on the received solution. These two PCR give two different result on gel electrophoresis. Then by a method called *graphical subtraction* on the gel plates the strands containing message could be obtained.

4.1.2 DNA Cryptosystem Using Random One-Time-Pad

In random one-time-pad cryptosystem first a codebook which consists of a large number of random keys are generated. The keys are used to encrypt the message. No key is used more than once. If the generated keys are truly random and no key is used more than once then this type of system is perfectly secure [26]. Gehani et. al. gave two methods for implementing one-time-pad system in [5] which are described bellow.

DNA cryptosystem Using Substitution

For these system encryption occurs by substituting each plaintext DNA word by a corresponding DNA chiper word. The substitution is done by taking a long dna strand which is partitioned into different regions. Each region consists of three parts: 1) one sequence word, C_i from the set of chiper or codebook matching words, 2) one sequence word P_i from the set of plaintext words, 3) a stopper sequence. Then each WC complement of plaintext P_i is used as primer in polymerization. The stopper sequence stops the extension of the polymerization any further. The polymerization produces plaintext-chipertext pair, which is cut thereafter in between paintext word and chipertext word.

Application of Substitution one-time-pad in DNA Cryptosystem for 2D images

In [5] it has been shown how to use the above substitution one-time-pad for practical purpose. The method proposed in [5] encrypts a 2D image of pixel values 0 or 1. For this purpose a DNA chip is required. DNA chips contain an array of pixels. Actually, each pixel is optically addressable and there are techniques to grow known sequence of DNA strands at a given address. It is assumed that both sender and the receiver have copies of same DNA chip and knows the codebook.

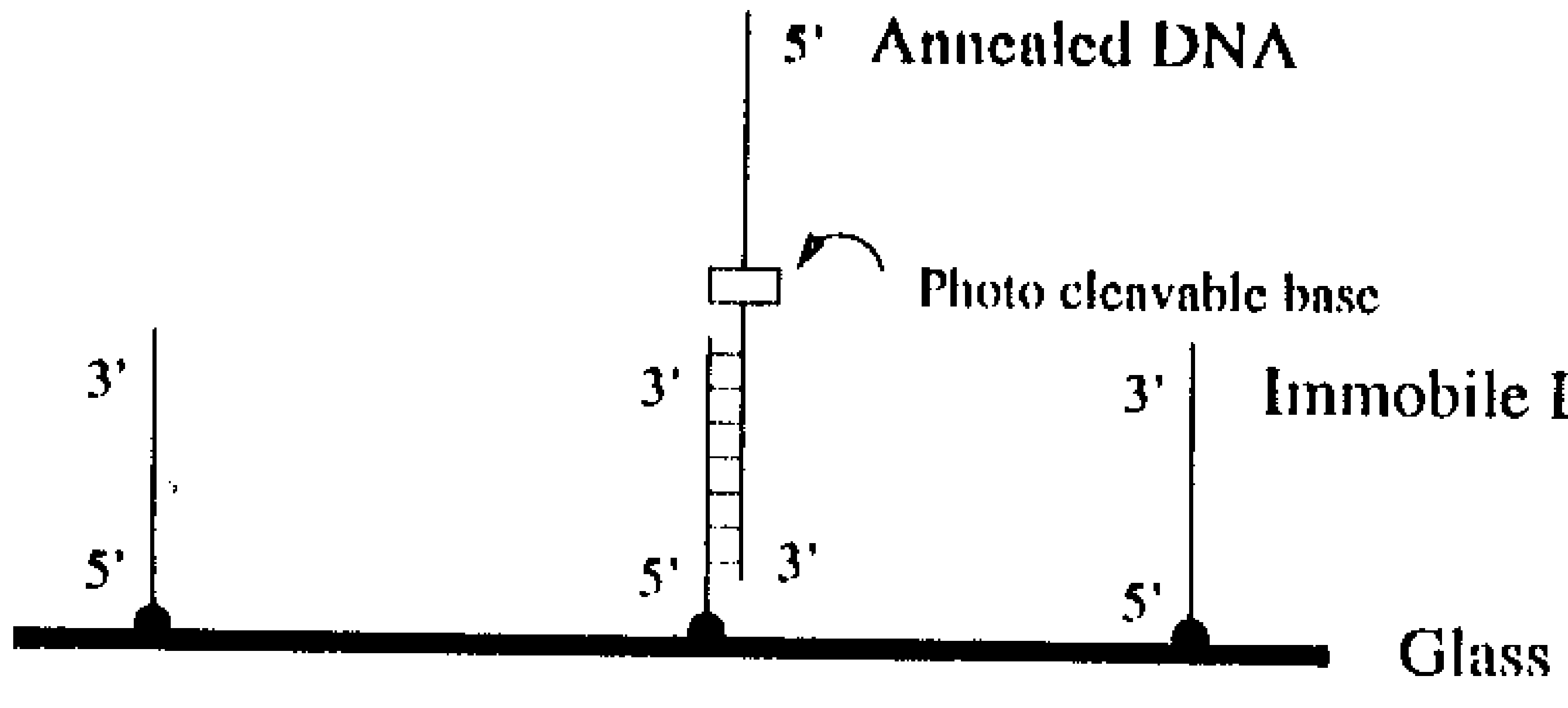

Figure 4.1: A DNA Chip.

For encryption, plaintext-chipertext word-pair is formed as in section 4.1.2. These are then annealed to their sequence complement at unique pixels on the DNA chip. The image is transformed to a photo mask which is transparent at pixels where the image pixel value is 1 and opaque at pixels where the image pixel value is 0. Following a light-flash of the mask protected chip, the word-pair beneath the transparent portion are cleaved(cut) at photo-liable positions. These cleaved portions are collected and sent to the receiver. The receiver does a PCR with the same codebook using the strands sent as primers to retrieve the word-pairs. These word-pairs are then allowed to anneal in the pixels of the receiver copy of the DNA chip. They will anneal at same location of the chip where from they were cleaved. If the strands are fluroscent, the receiver can ger back the image.

DNA Vernam One-Time-Pad

Vernam one-time-pad uses a sequence S of R uniformly distributed random bits. S serves as a codebook which is known to both sender and the receiver. The bits in S are used sequentially and no bits are used twice. Suppose the first L bits has already been used. Let $P_1, \ldots, P_n$ be the sequence of plaintext to be encrypted. The i^{th} bit in plaintext is encrypted to obtain chipertext C_i as follows:

$C_i = P_i \oplus S_{L+i}, \forall i = 1, \ldots, n.$

The decryption is done similarly,

$P_i = C_i \oplus S_{L+i}, \forall i = 1, \ldots, n.$

This is because $C_i \oplus S_{L+i} = P_i \oplus S_{L+i} \oplus S_{L+i}$.

The DNA cryptosystem in [5] implements the Xor using the Wang tile as described in [12]. The strand encoding the Xored values are sent to the receiver. The receiver again does the Xor with DNA Wang tiles and get backs the plaintext.

4.2 Cryptanalysis

In this section we review two attacks on Data Encryption Standard(DES) and an attack on public key cryptosystem NTRU.

DES is a cryptosystem based on Feastal Network. It takes a 64 bit plaintext M and a 56 bit key k to produce a 64 bit chipertext $C = DES(M, k)$. DES works in 16 rounds Figure 4.2. M_h and M_l are the higher and lower order 32 bits of M respectively.

The P-boxes permutes its input x to get output y. If $x = y$ then it is just a permutation. If $x > y$ then P-boxes outputs a subset of the bits in x in some order. If $x < y$ then the P-boxes replicates some of the bits in input. P-boxes simply change the order of the bits that arise during computation. When going through P-boxes there is no need to physically change the order of bits in the DNA strands. Thus, P-boxes are insignificant if we can keep track of the bits in DNA strand. The S-boxes in DES are boolean function that takes 48 bit input and gives a 32 bit output. Actually it divides 48 bits in 8 groups of 6 bits and to each group a boolean function is applied to get a 4 bit output.

Both the DNA computing attacks on DES are chosen chiper text attack. Both the attacks use only one given pair (M, C) of plaintext and chipertext. The idea of these attacks is to construct all ordered pair $(k, DES(M, k))$ encoded in DNA strands for a the given M and all possible key k. As the key is 56 bit long, there will be 2^{56} different strands of DNA encoding all the pairs. The next step is to find the key k for which the given chipertext C is equivalent to $DES(M, k)$.

4.2.1 Attack on DES by Boneh et. al.

In [4] Boneh et. al. used only extract, polymerize and PCR for breaking DES. To construct the ordered pairs $(k, DES(M, k))$ they followed the following steps:

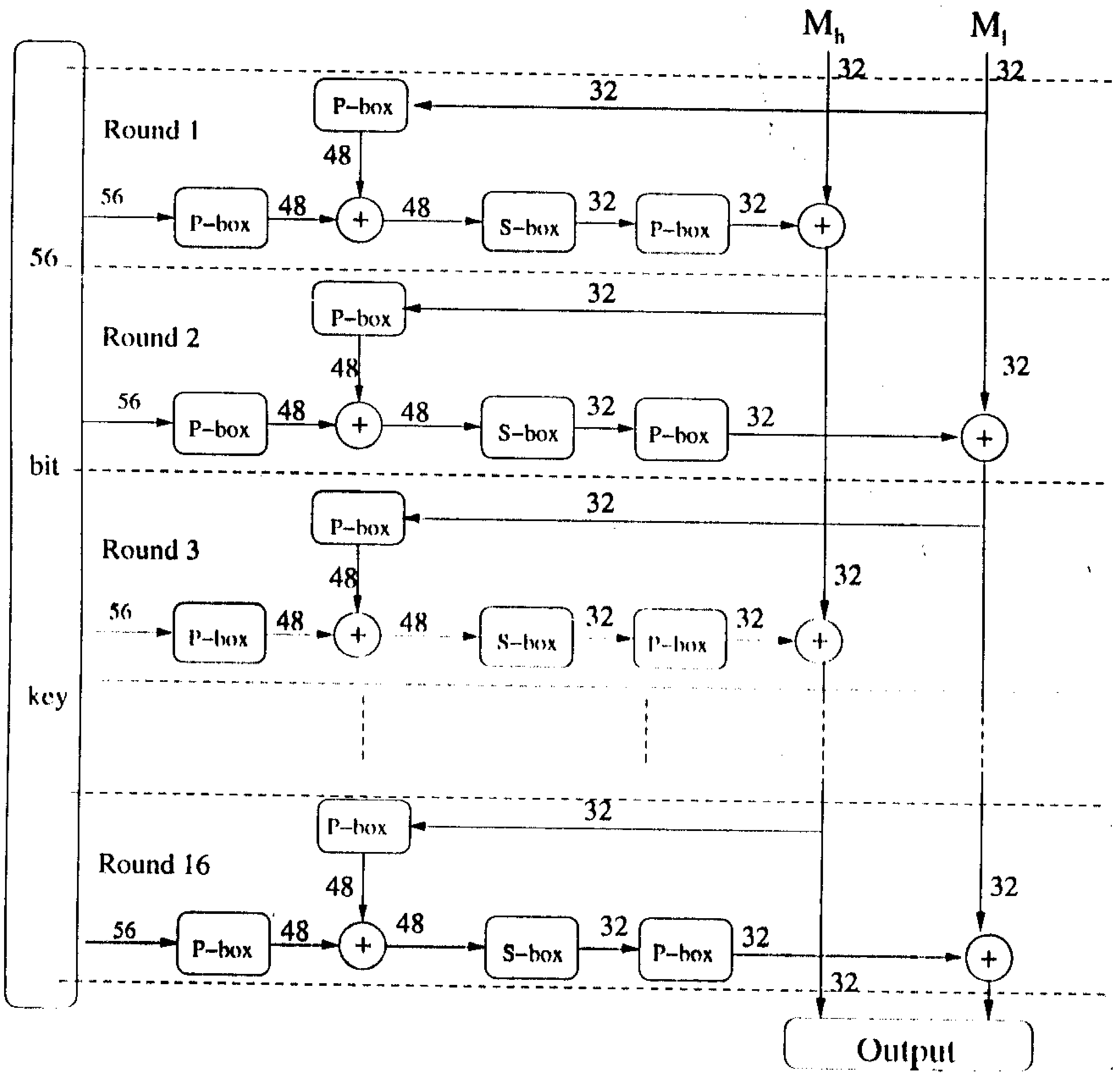

Figure 4.2: The DSE circuit.

1) Construct an initial solution encoding all 56 bit strings.

2) Evaluate DES circuit by tagging on their value. For example, if at some round the i^{th} and j^{th} bits are Xored then the Xor value is appended at the end of the string.

3) Evaluate the DES gates one by one. At the end the DNA strands will take the form $kxDES(M,k)$, where k is an encoding of the key k, x is the encoding of the intermediate calculation and $DES(M,k)$ is the encoding of the output.

Boneh et. al. calculated that the number of steps required for the evaluation of DES circuit is 916 extract operation which can be accomplished within 4 months. The also suggested some improvement over this method which requires 664 steps.

4.2.2 Attack on DES by Adleman et. al.

In [2] Adleman et. al. gave another method for breaking DES which uses sticker model(chapter 3). This method for has the added advantage that the DNA strands does not become too long making the molecule unstable. The method is also error resistant.

The initial test tube is a (579,56) library. The substrands in the memory complexes will be oligonucleotides of length 20. Thus the memory strands will be of length 11580 base pair. The procedure follows three steps:

1) Construct the initial (579,56) library encoding all 2^{56} keys.

2) On each memory complex, compute the chipertext obtained by encrypting the known plaintext by the key represented by the memory complex.

3) Select the memory complex whose cryptotext matches the known cryptotext and read its key.

An interesting feature of this method is that they used probabilistic technique to represent the key in the initial test tube. In this technique approximately 63% of the keys are represented on an average. This percentage can be increased upto 95% by taking three times the previous voulme. It was calculated that the number of steps required is 6655. With the help of a robotic arrangement efficient in performing the sticker model operations it will take approximately one day to break the system.

4.2.3 Attack on NTRU

The attack proposed in [18] to break the cryptosystem NTRU differs atleast in two respect from the previous two attacks: 1) it is an attack on a public key cryptosystem and 2) it is *not a brute force attack*. The main disadvantage of the attack is that it is not fully supported by physical experiments and the Wang tiles that are use for the attack to form 3D self-assembly has yet not been made practically. But from the work of Seeman in DNA nanostructure it seems feasible to construct these tiles.

Before describing the cryptosystem let us define the cyclic convolution product $*$ of two polynomials $f(x) = \Sigma_{i=0}^{N-1} f_i x^i$ and $g(x) = \Sigma_{i=0}^{N-1} g_i x^i$, both of degree N-1. The cyclic convolution product is defined as follows:

$$h(x) = f(x) * g(x) = \sum_{i=0}^{N-1} h_i x^i,$$

where

$$h_k = \sum_{i=0}^{k} f_i g_{k-i} + \sum_{i=k+1}^{N-1} f_i g_{N+k-i} = \sum_{\substack{i+j=k \ mod \ N, \\ i,j<N}} f_i g_j$$

Pelletier et. al. has proposed in [18] a method for computing the cyclic convolution product modulo some integer q.

The public key cryptosystem NTRU proposed in [9] is based on a ring $R = Z[x]/(x^N - 1)$, three integers (N, p, q), and four subset $\mathcal{L}_f$, $\mathcal{L}_g$, $\mathcal{L}_m$, $\mathcal{L}_\phi$ of R such that $gcd(p, q) = 1$, q is considerably larger than p. The multiplication in the ring is cyclic convolution product.

For key creation two polynomial $f \in \mathcal{L}_f$ and $g \in \mathcal{L}_g$ are chosen such that f has d coefficients of value 1 and $N - d$ coefficient of value 0, and g has coefficient in $\{0, \ldots, s-1\}$. First f_q and h is computed such that $f_q^{-1} \equiv f^{-1} mod\ q$ and $h \equiv f_q^{-1} * g$. The private key is f and the public key is h.

The chipertext e of $m \in \mathcal{L}_m$ is $e \equiv (r * h + m) mod\ q$, where r is a random element from $\mathcal{L}_\phi$.

The decryption could be done as follows:

$$m \equiv (f^{-1} mod\ p) * a\ mod\ q$$

where $a \equiv f * e\ mod\ q$ and the coefficients of a are chosen in the interval $-q/2$ to $q/2$.

The attack on NTRU is based on the *meet-in-the-middle* attack strategy. Meet-in-the-middle attack strategy for NTRU was proposed in [23] which guesses the private key f in the form $(f_1 \| f_2)$, where $(f_1 \| f_2)$ is the concatenation of f_1 and f_2. The attack involves computation of cyclic convolution product modulo some integer and it is this part that has been shown to be done efficiently using DNA Wang tiles which effectively breaks the system.

Chapter 5

DNA Algorithm for Breaking a Propositional Logic Based Cryptosystem

T. Head proposed an algorithm for 3-SAT. In [24], a modified algorithm which solves SAT in general is presented and it has been used for a cryptanalytic attack on a cryptosystem based on propositional logic.

5.1 The Cryptosystem

5.1.1 Description

The cryptosystem introduced by J. Kari [10] consists of two finite disjoint sets X and Y of propositional variables, α a truth value assignment of variable set X and p_0 and p_1 two propositional statements with the variables $X \cup Y$. The propositional statements p_0 and p_1 are such that p_0 is false and p_1 is true for every truth assignment of the variables in Y (with X having the same pre-assigned truth assignment α) i.e., for $\alpha : X \to \{true, false\}$ the same truth assignment chosen in advance and any arbitrary $\beta : Y \to \{true, false\}$, p_0 is false and p_1 is true.

The public keys are p_0 and p_1 and the private key is the secret assignment α of X.

5.1.2 Encryption Procedure

All messages are assumed to be in binary and the encryption is done in a bit-by-bit fashion. To encrypt a bit (0 or 1) any truth assignment β of the variables in Y is chosen first. With this assignment of Y we get propositional formulae p'_0 and p'_1 from p_0 and p_1 respectively, which contain variables from X only. To encrypt bit 0 (bit 1), p'_0 (p'_1 respectively) is taken and shuffled using the following set of rules:

$$\begin{array}{rcl@{\qquad}rcl}
\neg True & \rightarrow & False & \neg False & \rightarrow & True \\
(True \vee p) & \rightarrow & True & (False \wedge p) & \rightarrow & False \\
(True \wedge p) & \rightarrow & p & (False \vee p) & \rightarrow & p \\
(p \vee q) & \rightarrow & (q \vee p) & (p \wedge q) & \rightarrow & (q \wedge p) \\
(p \vee (q \vee r)) & \rightarrow & ((p \vee q) \vee r) & (p \wedge (q \wedge r)) & \rightarrow & ((p \wedge q) \wedge r) \\
(p \vee p) & \rightarrow & p & (p \wedge p) & \rightarrow & p \\
(p \vee \neg p) & \rightarrow & True & (p \wedge \neg p) & \rightarrow & False
\end{array}$$

Here $p \rightarrow q$ means any occurrence of p could be replaced by q (p, q, r are arbitrary statements). The shuffling does not change the truth value of the statements. A sequence of bits is encrypted as a sequence of propositional statements separated by a special marker.

5.1.3 Decryption Procedure

To decrypt the cryptotext, the secret truth assignment α of the variables of X is applied and 'true' ('false') is interpreted as '1' ('0' respectively).

The simplest method of breaking the cryptosystem is to find the secret assignment α. But this essentially means solving a particular instance of SAT problem, which is NP-complete. Tom Head has given a very simple and elegant algorithm based on circular DNA to solve the SAT problem [6].

5.2 DNA Plasmid and Molecular Operations

The fundamental data structure for this computation is an artificial plasmid, constructed as follows for a selected finite set of restriction enzymes. First, an artificial circular double-stranded DNA plasmid or Delphic plasmid containing a segment of the form

$$QC_1S_1C'_1C_2S_2C'_2\cdots C_jS_jC'_j\cdots C_{n-1}S_{n-1}C'_{n-1}C_nS_nC'_n\cdots$$

is constructed in such a manner that:

1) to every subsegment C_i, called a *site*, there corresponds a restriction enzyme R_i, that can cut the plasmid,

2) each subsegment S_i, called a *station*, is chosen in such a way that no other subsegment of the plasmid has the same base pair sequence,

3) the subsegment Q is a special sequence of base pairs such that no other subsegment of the plasmid has the same base pair sequence.

To present an algorithm for the cryptanalytic attack, we make use of molecular operations such as DELETE(station_ name), DROP(condition_ on_ station_ number) and DIVIDE(k) as in [6].

The key computational operation consists of the removal of one station, say S_i by the restriction enzyme R_i, from a (circular) plasmid followed by return of the resulting (linear) molecule to a (circular) plasmid again by adding ligase. This operation is done as follows: 1) R_i is added to the solution containing the plasmid. This cuts the plasmid into two linear pieces, a long piece and a short piece. The short piece is just the station S_i with a fragment of C_i at one end. 2) R_i and the short piece are removed from the solution. 3) Ligase is added to the solution to circularize the long piece to form (circular) plasmid again. This compound biochemical deletion process is expressed by DELETE(S_i) in the algorithm, where S_i is a station. If DELETE(S_i) is operated on a plasmid, the net change in the plasmid is a new plasmid that is identical with the original plasmid, except for the ommision of station S_i and the equivalent of one of the adjacent R_i sites C_i.

There are standard laboratory techniques such as gel separation, that allow DNA molecules to be separated into sets that have the same length as measured in bps. Such separation procedures allow the selection of those molecules that specify specific conditions, from a mixture of DNA molecules. Such a compound selection process is expressed by DROP(condition_ on_ molecule_ lengths) or equivalently DROP(condition_ on_ station_ number). Thus, DROP(condition_ on_ station_ number) will remove all plasmids that satisfy the condition.

The algorithm requires the contents of a single test tube to be divided into a number of equal parts in different test tubes and this is done by DIVIDE(k). In other words, in the procedure

```
DIVIDE($k$)
        DELETE($S_1$)
        DELETE($S_2$)
        ......
        ......
        DELETE($S_k$)
UNITE
```

the plasmids are divided into k new test tubes. Then in the first test tube, S_1 is deleted, in the second test tube S_2 is deleted, in the third test tube S_3 is deleted and so on. These steps can be done in parallel or in any desired sequential order. The contents of these k test tubes are then mixed into a single new test tube. So, at the end of the above procedure there will be no plasmid containing all stations S_i, $i = 1, 2, \dots, k$.

5.3 The Cryptanalytic Attack

The idea of the attack is as follows: 1) Generate the set Γ of all truth assignments of X that make p_0 false and p_1 true, irrespective of any truth assignment of Y. 2) Generate the set Γ' of all truth assignments of X that satisfy the propositional formula P_i, the encryption of the i^{th} bit of the plaintext. 3) Find $\Gamma \cap \Gamma'$. If P_i is derived from p_0 with some truth assignment of Y, then the truth assignments of X that satisfy P_i must not match with any of the truth assignments in Γ. This is because all the assignments in Γ make p_0 false whatever may be the truth assignment of Y. Similarly, if P_i is derived from p_1 then at least one assignment, namely α, must be present in both the sets Γ and Γ'. Thus, $\Gamma \cap \Gamma' = \emptyset$ implies P_i is the encryption of 0 and $\Gamma \cap \Gamma' \neq \emptyset$ implies P_i is the encryption of 1.

5.3.1 Construction of Initial Solution

With the public keys p_0 and p_1, we form propositional formula

$$P' = \neg p_0 \wedge p_1.$$

Let P be the conjunctive normal form (CNF) of the propositional statement P'. Let $x_1, x_2, ..., x_m$ be the variables in P from the set X and $\neg x_1, \neg x_2, ..., \neg x_m$ be their negations. With each of these $2m$ literals we associate a station in the plasmid. Let S_i be the station associated with variable x_i. If S_i is the station associated with *literal* x_i, then $\neg S_i$ denotes the station associated with $\neg x_i$. We construct a plasmid that contains all the $2m$ stations and a special station S. Let C be the site attached at both ends of S and R be the restriction enzyme corresponding to site C. *We will always start with a single test tube containing several* ***copies*** *of the* ***same plasmid.*** It is to be noticed that the base pair sequence of each plasmid is a circular permutation of the base pair sequence of another. We call this the initial solution.

5.3.2 Procedure 1

The following algorithm produces the set Γ, of all assignments of X that satisfy P, irrespective of any assignment of Y.

```
   1) Take some initial solution.
   2(a) Remove contradictions.

for $i = 1$ to $m$
{
  DIVIDE(2)
       DELETE($S_i$)
       DELETE($\neg S_i$)
  UNITE
}

   2(b){DROP(number_of_stations_associated_with_literals$\neq m$)}
   3) Let $s$ be the number of clauses in the CNF $P$.

for $i = 1$ to $s$
{
  Let $r_{i_1} \vee r_{i_2} \vee \cdots \vee r_{i_q}$ be the $q$ disjunctions in the $i^{th}$ clause.
  Let $r_{x_1}, r_{x_2}, \cdots, r_{x_t}$ be all the literals from $X$ appearing in it,
  i.e., $\{r_{x_1}, r_{x_2}, \cdots, r_{x_t}\} = \{r_{i_1}, r_{i_2}, \cdots, r_{i_q}\} \cap \{x_1, x_2, ..., x_m, \neg x_1, \neg x_2, ..., \neg x_m\}$.
  Let $S_{r_k}$ be the station associated with $r_{x_k}$, $k = 1, ..., t$.
  DIVIDE($t$)
       for $k = 1$ to $t$
       {
         DELETE($\neg S_{x_k}$)
       }
  UNITE
}

   4) DROP(number_of_stations_associated_with_literals$\neq m$ )
```

At the end of Procedure 1, the test tube contains plasmids corresponding to each element of Γ. The assignments are present in the form of stations of the plasmids. Call this test tube

$T1$. The restriction enzyme R is added to $T1$ so that the station S gets deleted, linearizing the (circular) plasmids. The single strands of the form

$$Q \cdots C_{i_1} S_{i_1} C_{i_1} \cdots C_{i_2} S_{i_2} C_{i_2} \cdots\cdots C_{i_m} S_{i_m} C_{i_m} \cdots$$

are extracted, with primer $\overline{Q}$ and the complementary parts discarded.

5.3.3 Procedure 2

Let P_i' be the formula encrypting the i^{th} bit and P_i its CNF form. Procedure 2 uses same initial solution and is similar to Procedure 1 but produces the set Γ' of all the assignments that satisfy P_i.

```
    1) Take some initial solution.
    2(a) Remove contradictions.

for $i$= 1 to $m$
{
 DIVIDE(2)
      DELETE($S_i$)
      DELETE($\neg S_i$)
 UNITE
}
    2(b){DROP(number_of_stations_associated_with_literals$\neq m$)}
    3) Let $s$ be the number of clauses in CNF $P_i$.

for $j$ = 1 to $s$
{
  Let $r_{j_1} \vee r_{j_2} \vee \cdots \vee r_{j_t}$ be the $t$ disjunctions in the $j^{th}$ clause of $P_i$
  and let $S_{j_k}$ be the station associated with $r_{j_k}$, $k = 1, ..., t$.
  DIVIDE($t$)
      for $k$ = 1 to $t$
      {
        DELETE($\neg S_{j_k}$)
      }
  UNITE
}
    4) DROP(number_of_stations_associated_with_literals$\neq m$).
```

The resultant after Procedure 2, called test tube $T2$, contains plasmids corresponding to each element of Γ'. The restriction enzyme R is added to $T2$. This deletes station S and the (circular) plasmids in $T2$ are linearized. With primer Q, complementary parts of the strands in $T2$ are extracted, discarding the other parts. Now, T2 contains linear single strands of the form

$$\overline{Q} \cdots \overline{C_{i_1}}\ \overline{S_{i_1}}\ \overline{C_{i_1}} \cdots \overline{C_{i_2}}\ \overline{S_{i_2}}\ \overline{C_{i_2}} \cdots\cdots \overline{C_{i_m}}\ \overline{S_{i_m}}\ \overline{C_{i_m}} \cdots$$

5.3.4 Breaking the Cryptosystem

The final step in breaking the cryptosystem is to show that it is possible to decide whether P_i encodes bit 1 or 0. This is done as follows. The contents of the test tubes $T1$ and $T2$ are poured into a fresh test tube $T3$. If presence of a well-formed, fully double-stranded DNA is detected, then P_i encodes bit 1, otherwise P_i encodes bit 0. To prove this, we note that since $T1$ and $T2$ are obtained from the same solution, if there is a common truth assignment in P' and P'', the corresponding plasmids are present in $T1$ and $T2$ at the end of Procedures 1 and 2. Hence, after linearization of the (circular) plasmids in $T1$ and $T2$, the corresponding linear strands are complementary. So, when the contents of $T1$ and $T2$ are mixed in $T3$, the complementary strands anneal to form well-formed, fully double-stranded linear molecules.

5.3.5 Time Complexity

Both in Procedure 1 and Procedure 2, removal of contradictions (step 2) takes $O(m)$ biosteps. If there are s_1 clauses in P and in each clause if there are a maximum of t_{max1} literals associated with X, then step 3 of Procedure 1 takes $O(s_1 t_{max1})$ biosteps. Similarly, step 3 of Procedure 2 also takes $O(s_2 t_{max2})$ biosteps, where s_2 is the number of clauses in P_i and t_{max2} is the maximum number of literals in each clause. Thus, the number of biosteps required for both the procedures is of the order of the size of the formula P or P_i as defined in [14]. As in [6, 14], we have not avoided the initial construction of exponential number of molecules that represents all possible truth assignments.

Chapter 6

Conclusion

In this project the various methods of DNA cryptology has been studied. It has been shown that in this paradigm the job of designing cryptosystems as well as attacking them could be well accomplished. The potential of this particular approach in cryptology is promising. A simple algorithm using circular DNA, to break the propositional logic based cryptosystem is presented. It is interesting since any attack to break the system will provide a method for breaking other cryptosystems as well. Another interesting feature is that it concerns all possible cryptanalytic attacks on the cryptosystem. For other cryptosystems, it is usual that only some particular cryptanalytic methods have been proved intractable. The algorithm itself is simple, and makes use of material that are easily available and well-known techniques and operations used in molecular computing.

But DNA computing as well as DNA cryptology is a very new field and not much work has been done in this field. Various questions regarding the implementation is yet to be resolved.

Bibliography

[1] L. M. Adleman. Molecular Computation of Solutions to Combinatorial Problems, *Science*, 266, November 1994, pp. 1021-1024.

[2] L. M. Adleman, P. W. K. Rothemund, S. Roweis and E. Winfree. On Applying Molecular Computation to the Data Encryption Standard, in L. F. Landweber and E. B. Baum, Eds., *DNA Based Computers, Proc. of the Second DIMACS Workshop, June 10-12, 1996*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 44, American Mathematical Society, 1999, pp. 31-44.

[3] D. Beaver. Factoring: The DNA-Solution, *Advances in Cryptology - Asiacrypt, 1994*, LNCS, Vol. 917, Springer-Verlag, pp. 419-423.

[4] D. Boneh, C. Dunworth and R. J. Lipton. Breaking DES Using a Molecular Computer, in R. J. Lipton, E. B. Baum, Eds., *DNA Based Computers, Proc. of a DIMACS Workshop, April 4, 1995*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 27, American Mathematical Society, 1996, pp. 37-66.

[5] Ashish Gehani, T. H. LaBean, and John H. Reif. DNA-based cryptography. in Winfree and Gifford, Eds., *Proceedings 5th DIMACS Workshop on DNA Based Computers*, held at the Massachusetts Institute of Technology, Cambridge, MA, USA June 14 - June 15, 1999. American Mathematical Society, 1999. vol. 54. , pages 233–249,

[6] T. Head. Circular Suggestions for DNA Computing, in A. Carbone, M. Gromov and P. Prusinkiewicz, Eds., *Pattern Formation in Biology, Vision and Dynamics*, World Scientific, Singapore and London, 2000, pp. 325-335.

[7] T. Head. Splicing schemes and DNA. *Lindenmayer Systems; Impact on Theoretical computer science and developmental biology*, pages 371–383, 1992.

[8] T. Head. Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, 49(6):737–759, 1987.

[9] J. Hoffstein, J. Pipher, J. H. Silverman. *Algorithmic Number Theory (ANTS III)*, Portland, OR, June 1998, J.P. Buhler (ed.), Lecture Notes in Computer Science 1423, Springer-Verlag, Berlin, 1998, 267-288.

[10] J. Kari. A Cryptosystem Based on Propositional Logic, *Machines, Languages and Complexity, 5th International Meeting of Young Computer Scientists, Czeckoslovakia, Nov. 14-18, 1988*, Eds. J. Dassow and J. Kelemen, LNCS 381, Springer, 1989, pp. 210-219.

[11] L. Kari. DNA Computing : Arrival of Biological Mathematics, *The Mathematical Intelligencer*, 19, No: 2 (Spring 1997), pp. 9-22.

[12] T. H. LaBean, E. Winfree, and J. H. Reif. Experimental progress in computation by self-assembly of DNA tilings. in Winfree and Gifford, Eds., *Proceedings 5th DIMACS Workshop on DNA Based Computers*, held at the Massachusetts Institute of Technology, Cambridge, MA, USA June 14 - June 15, 1999. American Mathematical Society, 1999. vol. 54. , pages 233 249, pp. 123–140,

[13] Andr Leier, Christoph Richter, Wolfgang Banzhaf, and Hilmar Rauhe. Cryptography with DNA binary strands. Biosystems, 57(1):13–22, 2000.

[14] R. J. Lipton. DNA Solution of Hard Computational Problems, *Science*, 268, 1995, pp. 542-545.

[15] C. Mao, T.H. LaBean, J.H. Reif, and N.C. Seeman. An algorithmic self-assembly. *Nature*, September 28, 2000.

[16] G. Paun, G. Rozenberg and A. Saloman. DNA Computing: New Computing Paradigms, Springer-Verlag, Berlin, 1998.

[17] G. Paun. Regular extended H systems are computationally universal. Journal of Automata, Languages, Combinatorics, 1(1):27–36, 1996.

[18] O. Pelletier, A. Weimerskirch, Algorithmic Self-Assembly of DNA Tiles and its Application to Cryptanalysis, GECC0-2002 N.Y. USA, pp. 139-146.

[19] D. Pixton. Regular splicing systems. Manuscript, 1995.

[20] D. Pixton. Regularity of splicing languages. *Discrete Applied Mathematics*, 69(1 2):101 124, August 1996.

[21] Dennis Pixton. Linear and circular splicing systems. *Proceedings of the 1st International Symposium on Intelligence in Neural and Biological Systems*, pages 181 188. IEEE, May 1995.

[22] S. Roweis, E. Winfree, R. Burgoyne, N. V. Chelyapov, M. F. Goodman, P. W. K. Rothemund, and L. M. Adleman. A sticker based model for DNA computation. *Proceedings of the Second Annual Meeting on DNA Based Computers*, held at Princeton University, June 10-12, 1996. pp. 1 27.

[23] J. H. Silverman, A Meet-In-The-Middle Attack on an NTRU Private Key. NTRU technical report 4, 1997.

[24] R. Siromoney, B. Das. DNA Algorithm for Breaking a Propositional Logic Based Cryptosystem, EATCS Bulletin, Feb 2003, pp. 70-78.

[25] Rani Siromoney, K.G. Subramanian, and V. Rajkumar Dare. Circular DNA and splicing systems. In A. Nakamura, M. Nivat, A. Saoudi, P.S.P. Wang, and K. Inoue, editors, *Proceedings of Parallel Image Analysis*, 2nd International Conference ICPIA '92, Ube, Japan,

21-23 Dec 1992., number 654 in Lecture Notes in Computer Science, pages 260–273, Ube, Japan, 1992. Springer Verlag, Berlin, Heidelberg, New York, ISBN 3-540-56346-6.

[26] D. R. Stinson. Cryptography: Theory and Practice. *CRC Press Inc.* 1995.

[27] C.T. Taylor, V. Risca, and C. Bancroft. Hiding messages in DNA microdots. *Nature*, 399:533–534, 1999.

[28] H. Wang. Proving Theorems by Pattern Recognition. II. *Bell System Technical Journal.* vov. 40. 1961, pp. 1-42.

[29] E. Winfree. On the computational power of DNA annealing and ligation. Eric B. Baum and Richard J. Lipton, Eds., *DNA Based Computers, volume 27 of DIMACS: Series in Discrete Mathematics and Theoretical Computer Science.* ISSN 1052-1798. American Mathematical Society, 1996, ISBN 0-8218-0518-5. Also known under the working title DNA Computing.