Technical Report No. 2007-539

School of Computing
Queen's University
Kingston, Ontario, Canada

Molecular Codebreaking and Double Encoding

Cameron D. McKay*, Joslynn G. Affleck†,
Naya Nagy, Selim G. Akl‡, Virginia K. Walker§

August 21, 2007

Abstract

We have evaluated the practical feasibility of implementing part of a molecular codebreaker, a DNA computer that uses a known-plaintext attack to recover an encryption key. Molecular biology techniques such as ligation, gel electrophoresis, polymerase chain reaction (PCR), and graduated PCR were found to be feasible but currently have some limitations such as error accumulation. A so-called "error resistance technique" of double encoding, where bits are encoded twice in a DNA strand, was also designed and attempted. Although the double encoding implementation was not completely successful, several important issues associated with using ligation for double encoding were identified, such as encoding adaptation problems, strand generation penalties, strand length increases, and the possibility that double encoding may not reduce the number of false negatives.

1 Introduction

DNA computers offer inherent parallelism with remarkable energy efficiency, information density, and can be as fast as modern silicon-based computer [2, 8, 7, 9, 13, 14]. The first DNA computer, devised by Adleman [2] was used to solve the directed Hamiltonian path problem. The Lipton and sticker-based computers [11, 15] were designed to tackle more general problems, with the former showing promise at least for computations with up to 60 variables and the latter exhibiting potentional even in partial implementations [8, 7]. Surface-based DNA computers [12, 13, 17] have been shown to reduce the amount of DNA lost in molecular biology purification steps but are subject to lower reaction kinetics since the DNA is immobilized. The hairpin DNA computer makes use of intrastrand hybridization (hairpin formation [16]), offering a promising new DNA

*Queen's University is acknowledged for financial support of Cameron D. McKay.
†NSERC supported Joslynn G. Affleck with a postgraduate scholarship.
‡NSERC supported Selim G. Akl with a Discovery Grant.
§NSERC supported Virginia K. Walker with a Discovery Grant.

computing approach. Innovations such as gel-based methods [8, 7] have not yet been embraced because of the necessary specialized equipment and an efficiency that presently cannot compete with inexpensive silicon-based computers. Most of these devices suffer from slow human-assisted steps for the biochemical reactions, but biomolecular automata [4] use proteins that rapidly recognize and hydrolyze certain DNA sequences (restriction enzymes) and join (ligases) appropriate strands allowing up to 10^{19} transitions per second. These and similar machines [1, 3] are perhaps the most promising DNA computers yet devised, but even these cannot theoretically perform all the computations possible *in silico*, and until now have only been validated for a simple case involving two states. Thus, DNA computing is in its infancy and it is uncertain if these computers will have general applicability in the future or if they will be required only for a limited number of specialized tasks.

One such task could be the breaking of encryption schemes, such as the widely-used but relatively insecure U.S. Data Encryption Standard (DES) that encrypts 64-bit messages using a 56-bit key. The use of DNA computers to solve such problems was described by Boneh *et al.* [5]. In their paper they described a theoretical, brute force attack to recover the key used to map a known-plaintext to a known-ciphertext. This exhaustive approach on a DNA computer has not yet been demonstrated in the laboratory. In this paper we attempt to use a much simpler, proof-of-concept version of the Boneh *et al.* program. The new program, designated Molecular Codebreaker 1 (MCB-1), was designed to have proof-of-concept simplicity in that it uses only 2-bit plaintexts, 2-bit keys and 2-bit ciphertexts. Furthermore, instead of full DES encryption, MCB-1 uses the principal operation performed by the DES algorithm: the exclusive disjunction, also known as the exclusive-OR operation, and denoted XOR.

In order to explore the feasibility of using a DNA computer to break encryption, we evaluated the feasibility of using ligation, gel electrophoresis, PCR, and graduated PCR (GPCR) in MCB-1. We also began some preliminary experiments using magnetic bead separation with MCB-1. All of these molecular biology techniques appeared to be suitable for use in a present-day real-world implementation of MCB-1. We also evaluated using a ligation-based double encoding technique that was to act as an error correction mechanism for MCB-1. Although the double encoding hypothesis could not be tested, our experiments yielded important observations regarding the use of ligation to implement double encoding. These observations are discussed in detail in Section 3.

The remainder of this paper is organized as follows: Section 2 covers the structure and implementation of MCB-1. It also covers the experiments we carried out and their results. Section 3 discusses our research into double encoding, particularly the different ways double encoding may be implemented in MCB-1. It also contains further experimental results and observations. The paper concludes with our conclusions and suggested future work in Section 4.

2 Molecular Codebreaking

As with the DES-breaking algorithm, the goal of MCB-1 is to determine the key K used to map a known-plaintext P to the known-ciphertext C. To do this, the following steps are followed:

1. The known-plaintext P is fixed to P_1P_2 where P_1 and P_2 are 0 or 1.
2. All keys are generated resulting in the set $K_1K_2 = \{00, 01, 10, 11\}$.
3. The ciphertexts are generated and appended, resulting in the set $K_1K_2T_1T_2$ where $T_i = K_i \oplus P_i$ for all $i \in \{1, 2\}$.
4. The known-ciphertext C is fixed to C_1C_2 where C_1 and C_2 are 0 or 1.
5. Any string from $K_1K_2T_1T_2$ where $T_i = C_i$ for all $i \in \{1, 2\}$ is separated to its own group. The key portion K_1K_2 is the key used to map P_1P_2 to C_1C_2.

Step 1 is a preprocessing step. Step 2 is the key generation step. Step 3 is the ciphertext generation step. Steps 4 and 5 are the key recovery steps.

2.1 Encoding Bits in DNA

To manipulate bits using molecular biology techniques, binary strings are encoded as DNA sequences. Each string is comprised of 4 bits: 2 bits for the key and 2 bits for the ciphertext. Since each of the 4 bits has two states (2×4), there are 8 oligonucleotides (oligos) required to encode the 4 bits in DNA. In addition to bit oligos, the DNA encoding of the 4-bit strings contains separator oligos. The 4-bit DNA encoding, with separators, has the following mathematical representation

$$S_0B_1(x_1)S_1B_2(x_2)S_2B_3(x_3)S_3B_4(x_4)S_4 \tag{1}$$

where x_i is 0 or 1, B_i is a bit oligo, and S_i is a separator oligo. The separators are used as templates for *Taq* polymerase so that the bits can be amplified using a polymerase chain reaction (PCR) and thus detected. The bit and separator oligo sequences used are listed in Table 1 and Table 2, respectively. The oligos are 20-40 base pairs (bp)[1] in length. In order to ensure that all oligos were distinct: (1) each nucleotide in an oligo sequence was chosen at random and (2) no two oligos share a substring of length 10 (for the 20-bp oligos), 15 (for the 30-bp oligos) or 20 (for the 40-bp oligos).

Connector oligos are designed to join bit and separator oligos together and are named for the two sequences connected (see Table 3). For example, a connector between S_1 and $B_2(0)$ is denoted $S_1 \leftrightarrow B_2(0)$. If S_1 and $B_2(0)$ were 20-bp, then the connector is designed to have a complementary sequence of the

[1]The number of base pairs (bp) in a DNA strand is the number of nucleotides in that strand. For example, the strand `GATTACA` has 7 nucleotides, so it is 7-bp long.

Bit	*Sequence*
$B_1(0)$	`GGTCGAGTGAGCCTGGTCACACAGCAGCGC`
$B_1(1)$	`GTGTTTGAGGTTCTATTTTGCCCACTAGAC`
$B_2(0)$	`TTACGGCCTTTGGATCATTTGTTTACTCGT`
$B_2(1)$	`CTGCAACTACTGAGTATTCCTGAGCCGGCA`
$B_3(0)$	`CTCTCGACTGCTACGCACGA`
$B_3(1)$	`TACTACTGCTATGTACTGCT`
$B_4(0)$	`CTGTAGTCGCAGTGCTCTCG`
$B_4(1)$	`AGATGTCGACGAGAGACATA`

Table 1: MCB-1 bit oligo sequences for the 8 states of 4 bits.

Separator	*Sequence*
S_0	`TCTGGTCACCTCCTCCTTCTCTGCCTACGA`
S_1	`TTCCACGCCGCAGAGCCATCTAGGTAGCTT`
S_2	`CGTACGTTCCCCTAACCGCATGCATTAGCT`
S_3	`GATGCTACTCGTCACGATAG`
S_4	`TCACGCTCAGTATACAGTGA`

Table 2: MCB-1 separator oligo sequences for 4 bits.

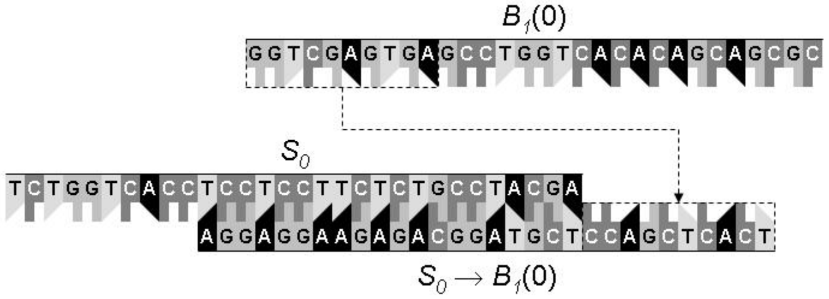
Connector	*Sequence*
$S_0 \leftrightarrow B_1(0)$	`AGACCAGTGGAGGAGGAAGAGACGGATGCTCCAGCTCACT`
$S_0 \leftrightarrow B_1(1)$	`AGACCAGTGGAGGAGGAAGAGACGGATGCTCACAAACTCC`
$B_1(0) \leftrightarrow S_1$	`CGGACCAGTGTGTCGTCGCGAAGGTGCGGC`
$B_1(1) \leftrightarrow S_1$	`AAGATAAAACGGGTGATCTGAAGGTGCGGC`
$S_1 \leftrightarrow B_2(0)$	`GTCTCGGTAGATCCATCGAAAATGCCGGAA`
$S_1 \leftrightarrow B_2(1)$	`GTCTCGGTAGATCCATCGAAGACGTTGATG`
$B_2(0) \leftrightarrow S_2$	`ACCTAGTAAACAAATGAGCAGCATGCAAGG`
$B_2(1) \leftrightarrow S_2$	`ACTCATAAGGACTCGGCCGTGCATGCAAGG`
$S_2 \leftrightarrow B_3(0)$	`ACGTAATCGAGAGAGCTGAC`
$S_2 \leftrightarrow B_3(1)$	`ACGTAATCGAATGATGACGA`
$B_3(0) \leftrightarrow S_3$	`GATGCGTGCTCTACGATGAG`
$B_3(1) \leftrightarrow S_3$	`TACATGACGACTACGATGAG`
$S_3 \leftrightarrow B_4(0)$	`CAGTGCTATCGACATCAGCG`
$S_3 \leftrightarrow B_4(1)$	`CAGTGCTATCTCTACAGCTG`
$B_4(0) \leftrightarrow S_4$	`TCACGAGAGCAGTGCGAGTC`
$B_4(1) \leftrightarrow S_4$	`CTCTCTGTATAGTGCGAGTC`

Table 3: MCB-1 connector oligo sequences for 4 bits.

last 10 bp of S_1 and the first 10 bp of $B_2(0)$. Collectively, the bit, separator and connector oligos are referred to as the encoding set. The encoding set uses 30-bp oligos (the first two bits), as well as 20-bp oligos (last two bits). Although longer than required for a 4-bit problem, the 30-bp oligos allow for the possibility of theoretically scaling to 300 bits or more.

2.2 Key Generation

In the key generation step of MCB-1, all possible keys are generated. The keys are generated using the first two bits of the encoded DNA string. That is, the $S_0B_1(x_1)S_1B_2(x_2)S_2$ part of Equation 1. By taking advantage of the natural affinity that complementary nucleotides have for one another (i.e. Watson-Crick complementarity), ligation reactions are used so that the oligos that make up $S_0B_1(x_1)S_1B_2(x_2)S_2$ form the bit strings 00, 01, 10, and 11 in DNA (see Figure 1 for an example). The new, longer DNA strands that result from the ligation of the smaller oligos are called *keyligos*.



$B_1(0)$
GGTCGAGTGAGCCTGGTCACACAGCAGCGC
S_0
TCTGGTCACCTCCTCCTTCTCTGCCTACGA
AGGAGGAAGAGACGGATGCTCCAGCTCACT
$S_0 \rightarrow B_1(0)$

Figure 1: A figure depicting how the S_0 and $B_1(0)$ connect together. In the figure, S_0 and $S_0 \leftrightarrow B_1(0)$ have annealed to form hydrogen bonds because they share 20 complementary nucleotides. Moreover, since the first 10 nucleotides of $B_1(0)$ are complementary to the last 10 nucleotides of $S_0 \leftrightarrow B_1(0)$, they too will anneal along the area indicated. The new oligo will be 60-bp long.

Once the ligation reaction has been completed, the solution is analyzed by gel electrophoresis to verify that the keys formed correctly. Following gel analysis, the keyligos are amplified using PCR with S_0 (forward primer) and $\bar{S_2}$[2] (reverse primer) as primers.

2.3 Ciphertext Generation

In the ciphertext generation step of MCB-1, the keys generated in the previous step are used to encrypt the known-plaintext. In order to generate the cipher-

[2] $\bar{S_2}$ is the Watson-Crick complement of S_2. For example, if S_2 = `GACTACAC` then $\bar{S_2}$ = `CTGATGTG`.

texts from the known-plaintext P and the keyligos, a 2-phase cycle is repeated for every state of every bit in the key for the molecular algorithm.

In the first phase, the keyligos encoding a bit state $B_i(x_i)$ (where i is 1 or 2 and x_i is 0 or 1) are separated from the other keyligos using biotin-streptavidin bead separation[3]. For example, if the bit state to be separated was $B_1(0)$, then the bead separation would place all keyligos encoding 00 and 01 in a separate test tube.

In the second phase of the cycle, the appropriate bit of the ciphertext is added to the separated keyligos. To determine the ciphertext bit, the separated bit state $B_i(x_i)$ is XORed with the first bit of the known-plaintext P. So, continuing with the example from the first phase, if $P = 10$ and the bit separated was $B_1(0)$ then $1 \oplus 0 = 1$, meaning that 1 must be added to the separated keyligos. Note that the calculation of the ciphertext bit is performed manually, without the use of DNA. The ciphertext bit is appended to the keyligos through the use of a ligation reaction. If the ciphertext bit is represented by $B_3(1)$ then, to append it to the separated keyligos, the oligos encoding $S_2 \leftrightarrow B_3(1)$ and $B_3(1)$ must be ligated to the separated keyligos. After the ciphertext bit is appended, the separated solution (now with an extra ciphertext bit) is merged back into the solution with the rest of the keyligos.

The first two cycles of the ciphertext generation step are shown in Figure 2. Since each bit has 2 states and there are 2 bits in a key, 4 cycles must take place in order to generate all possible ciphertexts for a known-plaintext. The keyligos with ciphertexts attached to them are called *cryptligos*.

2.4 Key Recovery

In the key recovery step of MCB-1, the key used to encrypt the known-ciphertext is recovered. To recover the key, the cryptligos that encode the known-ciphertext (the *known-cryptligos*) are separated from the remaining cryptligos using bead separation. The attached keyligos are then read using graduated PCR (GPCR)[4].

The known-cryptligos are separated from the rest of the cryptligos using iterative bead separation operations. Consider an example where the known-ciphertext is 01. In order to separate all cryptligos that encode the known-ciphertext, a bead separation must be performed on the first bit of the ciphertext. Since the first bit of the ciphertext is located at the third bit in a cryptligo, the bead separation operation is made to separate at $B_3(0)$. As a result, the cryptligos that encode the ciphertexts 00 and 01 are placed in a separate tube. Next, a bead separation is performed on the test tube containing 00 and 01. Since the second bit of the ciphertext is 1, the bead separation operation is made to separate at $B_4(1)$. This separates all cryptligos that encode the ciphertext 01 in a separate tube. Next, the key encoded by the keyligos that are

[3]Biotin-streptavidin bead separation (described by Adleman [2] and Kaplan [10]) is used to separate DNA strands with a desired nucleotide sequence from those without it.

[4]GPCR is the technique that Adleman used in his first experiment to read graphs from specially encoded DNA. GPCR works by selectively amplifying substrings of DNA. See Adleman's paper [2] for further details.

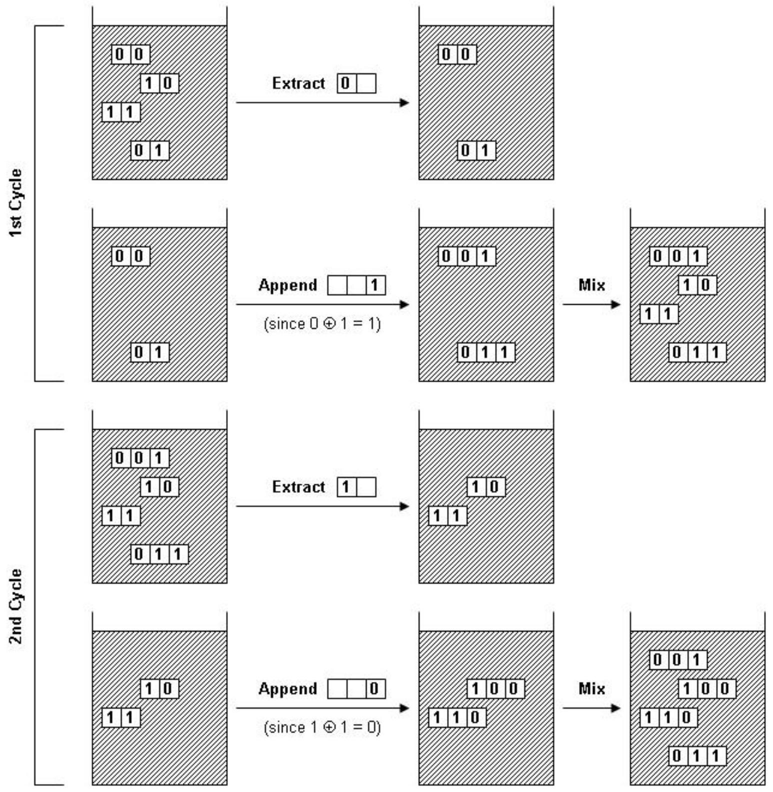


Figure 2: A diagram showing the first two cycles of the ciphertext generation step of MCB-1. In the example, the plaintext P is 10. In the first phase of the first cycle, the keyligos containing bit $B_1(0)$ (that is, the first bit with state 0) are separated from the rest of the keyligos. In the second phase, the ciphertext bit is calculated and appended to the separated keyligos. At the end of the cycle, the separated oligos are mixed back with the rest of the keyligos. The second cycle proceeds in the same manner as the first cycle, with the exception being that the separation occurs on bit $B_1(1)$ instead of $B_1(0)$.

attached to the known-cryptligos is recovered by using GPCR.

2.5 Results and Discussion

2.5.1 Ligation

In the key generation and ciphertext generation steps of MCB-1, a ligation reaction is used to either create keys or attach ciphertexts. Since there is no single optimal ligation protocol, three protocols were tested: ADL-4, ADL-12 and JC-12. Each of these protocols is briefly described below.

Protocol ADL-4 was used by Adleman in his first DNA computing experiment [2]. The protocol specified that all the necessary oligonucleotides should be mixed together along with ligase and left at room temperature for 4 hours. Protocol ADL-12 is identical to ADL-4 except that the reaction continues for 12 hours (*i.e.* overnight) in an attempt to maximize the ligation products. In contrast to Adleman's protocol, Protocol JC-12 has two steps. In the first step, the annealing step, all the necessary oligos are mixed together in a test tube along with the appropriate buffer and $MgCl_2$. The test tube is placed in an 80°C bath for 3 minutes then left for 1 hour to cool to room temperature. This allows the strands to anneal together first before they are ligated. In the second step, the ligation step, ligase is added and the solution is left for 12 hours. This step covalently bonds together the oligos annealed in the previous step.

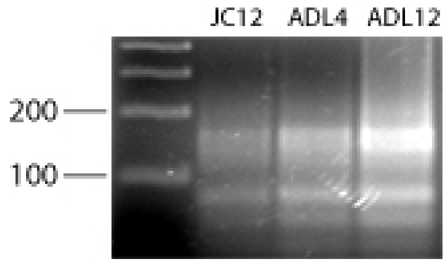


Figure 3: The ligation products formed using Protocols JC-12, ADL-4 and ADL-12. The first lane (far left) contains the base pair ladder. The second lane contains the keyligos made using JC-12, the third lane contains the keyligos made using ADL-4, and the fourth lane are the keyligos made using ADL-12. A 4% agarose gel stained with ethidium bromide is shown.

Experiments conducted with each of the three protocols showed appropriately-sized ligation products (see Figure 3) in the desired 150-bp area. ADL-4, the fastest of the three protocols, fared remarkably well with a bright band at 150-bp as well as two secondary bands below the 100-bp marker. ADL-12 produced the same bands as ADL-4, although the bands produced by ADL-12 were brighter, indicating more product. JC-12, the most labour-intensive of the three protocols, produced two faint bands at 150-bp and just below the 100-bp marker.

The most interesting result of the testing was the difference between the brightness of the bands produced by ADL-12 and JC-12. It appears that allowing the oligos to anneal before ligating (as is in the case with JC-12) results in a fewer number of keyligos being formed. It should also be noted that all three protocols produced secondary bands. However, because the 150-bp band is cut out of the agarose gel anyway, secondary products are of little concern. Thus, since ADL-4 was the fastest of the three protocols and still produced acceptable bands in the 150-bp range, it was chosen as the ligation protocol for MCB-1.

2.5.2 Gel Electrophoresis

Gel electrophoresis is used at some point in almost any DNA computer. In MCB-1, gel electrophoresis is used to confirm the computation is proceeding, to purify DNA, and to output bit strings.

Although gel electrophoresis has been a standard tool in molecular biology for quite some time, it is not error-free. Typically, none of the process is automated. That is, a researcher must mix the agarose and heat it, physically pour the liquid gel, transfer it to the electrophoresis apparatus and attach a power source. Because of this reliance on manual labour, consistency from gel to gel is difficult to guarantee. For instance, if the gel is not poured level the migration of the DNA may be altered. If the wells are not deep enough, then they may get overfilled and leave DNA behind. Other failures include DNA that runs at different speeds depending on the lane, lanes that do not appear parallel to the gel, DNA bands that bend, and air bubbles that may be trapped within the gel causing inhomogeneous polymerization of the gel [10].

During the course of the experiments, it became obvious that gel electrophoresis failures can have negative consequences in a DNA computing context. For instance, if a gel fails then, at best, the DNA can be cut out and recovered or, at worst, the DNA is lost. If the DNA is cut out and recovered, then the time spent preparing and running the gel (2 or more hours) is lost. If the DNA cannot be recovered (very rare) and the DNA computation is at an advanced stage, many days of work may be forfeited. Thus, just as with a silicon computer, DNA computers have a risk of “crashing”.

2.5.3 PCR

PCR is used in two steps of MCB-1. It is used to duplicate and purify the keyligos in the key generation step, and it is used to output the recovered key in the key recovery step. Unlike gel electrophoresis, most of the PCR reaction is automated by a machine known as a *thermal cycler*, which controls the temperature according to a preprogrammed schedule.

As with ligation, there is no definitive protocol for PCR. However, at least two protocols have been used in the DNA computing literature. PCR-A, used by Adleman in [2], specifies a temperature cycle of 94°C for 15 seconds and 30°C for 30 seconds, for a total of 35 cycles. PCR-K, used by Kaplan in [10], specifies a temperature cycle of 94°C for 30 seconds, 55°C for 30 seconds, and 72°C for

30 seconds, for a total of 35 cycles. Since both Adleman and Kaplan reported success using their respective protocols, it was decided that PCR-A would be used for all the experiments. This was decided based on the fact that PCR-A took less time in the thermal cycler than PCR-K and thus could be performed faster.

Of all molecular biology techniques used in the experiments, PCR was the least difficult to perform. Since the majority of the process is handled by a thermal cycler, very little manual effort is required beyond mixing the necessary reagents in a test tube and remembering which test tube contains which reaction. Furthermore, the high level of automation results in more consistent results than the more labour-intensive experiments like gel electrophoresis.

2.5.4 Graduated PCR

In his breakthrough paper [2], Adleman detailed an experimental paradigm that could solve the directed Hamiltonian path problem using a DNA computer. However, in order to ensure that the graph he received at the end of his DNA computation was indeed the correct graph, Adleman needed a technique for reading graphs encoded in DNA. The technique he used was graduated PCR (GPCR), a technique that uses PCR to selectively amplify specially encoded DNA substrings. In MCB-1, GPCR is used as part of the key recovery step. However, unlike Adleman, it is used to read bit strings instead of graph edges. This can lead to some ambiguous results (more on that below). In order to ensure that GPCR was a viable technique for reading the bit values of keyligos, it was decided that it would be tested in an idealized situation. Thus, GPCR was used to identify a 01 keyligo in a solution containing only 01 keyligos, which would occur only if the bead separation mechanism worked perfectly.

Since Adleman was the first researcher to use GPCR, his protocol [2] was used. The gel image of the GPCR test can be found in Figure 4. As can be seen in the figure, the GPCR was successful. However, the results were not ideal. Performing GPCR on a single key is the best-case scenario and should ideally result in only two bands, one at 60-bp in the second lane (representing bit 0 at position 1), and one at 120-bp in the fifth lane (representing bit 1 at position 2). In practice, there were 3 bands in the fifth lane, with the 120-bp band fainter than the 100-bp band. Perhaps most worrisome was the existence of a faint 60-bp band in the third lane, which could lead to ambiguous GPCR readings. Some possible reasons for the existence of the 60-bp band in the third lane are contamination or non-specific binding during the PCR process.

That being said, the GPCR technique appears to be a viable means to read the values of keyligos. Since all the PCR reactions necessary for GPCR can be performed in parallel, the technique has a time complexity of $O(1)$. The main issue with GPCR is the unexplained bands in the second and fifth lanes. Performing GPCR tests on the remaining three keyligos may shed light on this, and is recommended as future work.

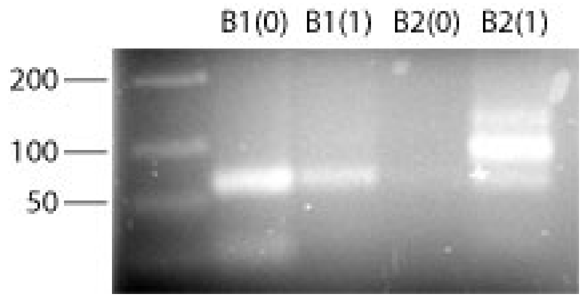


Figure 4: The GPCR test results attempting to read a 01 keyligo. The first lane (far left) contains the base pair ladder. The second lane, representing $B_1(0)$, is a clear band at 60-bp. This means that the first bit of the keyligo is 0. In the third lane, representing $B_1(1)$ there is a faint band at 60-bp, which should be ignored due to the presence of the brighter band in the second lane. In the fourth lane, representing $B_2(0)$, there are no bands. In the fifth lane, representing $B_2(1)$ there are three bands at 60-bp, 100-bp and 120-bp. Ideally, only the 120-bp band should be present. However, since there is a product in this lane, and none in the fourth lane, it can be assumed that the second bit of the keyligo is 1. A 4% agarose gel stained with ethidium bromide is shown.

2.5.5 Bead Separation

Bead separation is a molecular biology technique used in DNA computing for separating DNA strands based on their nucleotide sequences. In MCB-1, bead separation is used to separate DNA double-strands into single-strands[5], and then separate DNA single-strands based on their sequence. Preliminary work using bead separation to separate double-strands into single-strands was carried out in the laboratory.

The protocol used by us for separating DNA double-strands was a combination of the protocol used by Adleman [2] and the protocol included in the manufacturer's instructions. Bead separation was found to be a relatively straightforward experiment to perform. The only potential issue identified was with the frequent "washing" that took place. Washing entails placing a set amount of a solution in the test tube, moving the tube to a new spot in a magnetic tube rack, waiting 2-3 minutes, and then removing the solution (without accidentally taking the beads with it). This is repeated several times. The result is an operation that is tedious to carry out but also requires attention and precision, a combination that lends itself easily to human error.

Although extensive testing was not carried out, due to time constraints, using bead separation as the mechanism to separate DNA double-strands into single-

[5]It should be noted that separating double-strands into single-strands is not done exclusively by bead separation. Indeed, double-strands are first denatured (*i.e.* the bonds between the `A/T` and `C/G` are broken) into two single-strands, and then one of the two single-strands is removed using bead separation.

strands in MCB-1 appeared to be feasible. Since all DNA double-strands can be separated into single-strands in parallel, the operation can be performed in $O(1)$ time. However, separating DNA strands is only half of the bead separation operation. The second half of the operation, separating single-strands based on sequence, must also be tested for feasibility. More extensive testing using bead separation with MCB-1 has been left as future work.

2.5.6 Summary

MCB-1 was designed to test the basic functionality of the DES-breaking DNA computer described in [5]. In this respect, MCB-1 was a success. Using a 2-bit key and 2-bit ciphertext, in lieu of the original 56-bit key and 64-bit ciphertext, and an XOR gate, instead of the DES algorithm, allowed for the possibility of testing the molecular codebreaking concept with technology available today. It should be noted that using a DNA computer to break encryption has never before been attempted experimentally. Therefore, the research carried out in this paper represents an important first step toward one day using DNA computers in cryptanalysis. Although MCB-1 was not fully implemented, the results observed thus far indicate that MCB-1 is realizable now, with more complicated machines not too far in the future.

3 Double Encoding

DNA operations are *not* error free. An often cited example is the problem with separation by hybridization (*i.e. magnetic bead separation*. In the example, there is a test tube containing both good strands (those containing the desired DNA substring) and bad strands (the strands not containing the desired DNA substring). If, in each application of the bead separation operation, 99% of the good strands are removed, then the probability that a good strand remains at the end of 1,000 applications is $0.99^{1000} \approx 0.00004$. Thus, even with a 99% extraction rate, a bead separation-intensive algorithm that does not implement some sort of error correction will always fail [6].

The basic idea behind *double encoding* is to modify the way that binary strings are encoded in DNA in order to reduce the rate of false negatives[6]. In Lipton encoding, a binary string $x = x_1x_2\cdots x_n$ takes the form:

$$B_1(x_1)B_2(x_2)\cdots B_n(x_n) \tag{2}$$

where each $B_i(x_i)$ is a unique oligonucleotide for the bit at position i with value x_i. Thus, every $B_i(x_i)$ needs two oligonucleotides to represent it: one for $B_i(0)$ and one for $B_i(1)$. This encoding (Equation 2) will be referred to as *single encoding*.

Traditionally, false negatives in DNA computing are dealt with by increasing the amount of DNA strands in the solutions, thus increasing the probability

[6]False negatives occur when a desired strand remains unextracted.

that a good strand (that is, a strand containing the desired substring) will be extracted.

Double encoding, on the other hand, encodes the string $x = x_1 x_2 \cdots x_n$ twice, as in:

$$B_1(x_1)B_2(x_2)\cdots B_n(x_n)B_1(x_1)B_2(x_2)\cdots B_n(x_n) \tag{3}$$

That is, each $B_i(x_i)$ appears twice at separate positions in the strand. The belief is that since $B_i(x_i)$ appears twice in the DNA strand, it is more likely to be extracted than a strand that has only one $B_i(x_i)$.

A simple analysis [6] shows that double encoding is more likely to make tube-based DNA computing more resistant to false negative errors than doubling the amount of strands.

3.1 Implementation

There are three ways that double encoding could have been incorporated into MCB-1: the plaintext could have been double encoded (Equation 4), the ciphertext could have been double encoded (Equation 5), or both the plaintext and ciphertext could have been double encoded (Equation 6). The mathematical representation of the three encoding options (without separators or PCR markers) is shown below.

$$B_1B_2B_1B_2B_3B_4 \tag{4}$$

$$B_1B_2B_3B_4B_3B_4 \tag{5}$$

$$B_1B_2B_1B_2B_3B_4B_3B_4 \tag{6}$$

Since double encoding had hitherto not been attempted, it was decided that the initial implementation should be kept to a minimum. As a result, double encoding both the plaintext and the ciphertext was eliminated as a possibility. In order to choose between the two remaining possibilities, the following reasoning was applied. In MCB-1, 4 bead separation operations are performed on the plaintext portion of the DNA encoding, while only 2 are performed on the ciphertext portion. Double encoding is intended to make the bead separation operation more reliable. Thus, the plaintext portion of the encoding should be double encoded, since it undergoes twice as many bead separation operations as the ciphertext portion.

As a consequence of the decision to double encode the plaintext, the actual encoding used in MCB-1 needed to be modified. The original encoding

$$S_0B_1(x_1)S_1B_2(x_2)S_2B_3(x_3)S_3B_4(x_4)S_4 \tag{7}$$

was changed to

$$M_0S_0B_1(x_1)S_1B_2(x_2)S_2S_*S_0B_1(x_1)S_1B_2(x_2)S_2M_1B_3(x_3)S_3B_4(x_4)S_4 \tag{8}$$

where B_i is 0 or 1, S_i is a separator, and M_i is a PCR marker. The new encoding contained a special separator oligo (S_*) and two PCR markers M_0

and M_1. The S_* oligo allowed the first two bits of the string to repeat. The two PCR markers enabled the key portion of the string to be duplicated using PCR. For example, if S_0 and S_2 were used as primers instead of M_0 and M_1, then the PCR reaction would yield spurious results due to the existence of more than one S_0 or S_2 binding location.

In order to accommodate these changes to the DNA encoding, new oligos need to be generated. The new oligos can be found in Table 4[7].

Oligo	*Sequence*
S_*	`ATATATACGTGCATCAGCGTCAGCGTGCAG`
$S_2 \leftrightarrow S_*$	`GGATTGGCGTACGTAATCGATATATATGCA`
$S_* \leftrightarrow S_2$	`CGTAGTCGCAGTCGCACGTCAGACCAGTGG`
M_0	`TCTGAATGCGCTACTGTCTG`
M_1	`CGACCAATCGTTAGCACAAG`

Table 4: The additional separator and connector oligos used in the double encoded molecular program.

The encoding is not the only part of MCB-1 that needed to be modified in order to take advantage of double encoding. Indeed, the key generation step is changed significantly. Instead of generating all possible keys in a single ligation reaction, as is done in the *single encoded program* (SEP), the *double encoded program* (DEP) requires each key to be generated separately.

The reason the keys need to be assembled separately is due to the S_* oligo. In the SEP, the keys could be generated in parallel because they assemble in a non-ambiguous way. Unfortunately, in the case of the DEP, a non-ambiguous assembly cannot be guaranteed. If the key generation step of the DEP is carried out as a single ligation reaction (as is done in the SEP) then the following would occur. First, the single encoded keys would form: 00, 01, 10, 11. Second, the S_* oligo (represented as a +) would attach to the ends of some of the keys, allowing for the keys to repeat: 00, 00+, 01, 01+, 10, 10+, 11, 11+. Third, the keys would repeat. Unfortunately, the S_* at the end of 00+ cannot tell if it is connecting to 00 or 01 or 10 or 11. Thus, instead of yielding just the double encoded key 00+00, the reaction would also yield the 3 "mangled" keys 00+01, 00+10, and 00+11. This is unacceptable.

However, if each key is generated separately, the formation of "mangled" keys can be eliminated. Consider the case where a ligation reaction is carried out using only the oligos for the 00 key. First, a single encoded key would form, say 00. Second, the S_* oligo would attach to the end of some of the keys: 00, 00+. Finally, the keys would repeat, yielding only 00+00, since no other keys are present. Unfortunately, generating the keys separately in this fashion has an enormous penalty. The complexity for generating the n-bit keys one-by-one is $O(2^n)$. Conversely, the complexity for generating n-bit keys in parallel (as

[7]The S_* separator oligo, along with its two corresponding connector oligos, could be omitted in place of a single connector oligo between S_2 and S_0.

is done in the SEP) is $O(1)$. Thus, implementing double encoding in MCB-1 in this way increases the complexity of the key generation step from $O(1)$ to $O(2^n)$, a significant increase. The ciphertext generation and key recovery steps were unchanged for the DEP.

3.2 Results

The implementation of double encoding in the molecular program was unsuccessful. Specifically, the double encoded strands could not be made to form. Two different approaches to forming the strands were attempted. They are described below.

3.2.1 All-At-Once Approach

In the all-at-once approach, all the oligonucleotides that make up a particular double encoded key are added to a test tube and ligated. Although the oligo assembly pattern is not completely unambiguous (i.e. both S_* and P_2 can bind on the right side of S_2), the rationale is that the double encoded key will still form in sufficient quantities to allow for PCR amplification.

Unfortunately, this did not occur in the experiments, and, as a result, the all-at-once approach was discarded.

3.2.2 Half-and-Half Approach

The main perceived problem with the all-at-once approach was the potential for ambiguous assembly. The half-and-half approach is intended as a solution to that problem. Recall that a double encoded key is represented mathematically as:

$$P_0 S_0 B_1(x_1) S_1 B_2(x_2) S_2 S_* S_0 B_1(x_1) S_1 B_2(x_2) S_2 P_1$$

In the the half-and-half approach, the double encoded DNA strand is assembled in halves. One half consists of the $P_0 S_0 B_1(x_1) S_1 B_2(x_2) S_2 S_*$ portion, and the other half consists of the $S_0 B_1(x_1) S_1 B_2(x_2) S_2 P_1$. Each of these halves are ligated separately, giving two separate DNA strands. After being ligated separately, the two halves are ligated together in an additional ligation reaction.

The experimental work found that, although the two half strands appeared to form successfully, they refused to stick together after the final ligation reaction. Ultimately, the half-and-half approach had to be abandoned as well.

3.3 Discussion

Double encoding is intended to reduce the error associated with using hybridization as a central mechanism of a DNA computer. Indeed, the idea of double encoding is simple to grasp: offer two locations on a single strand for a DNA probe to bind to. In theory, this should offer better performance than having just a single location on twice as much DNA. However, the experiments failed to yield results to prove or disprove this hypothesis.

This failure can be attributed to the multitude of problems associated with using DNA in a computing context. As Kaplan [10] reported when attempting to repeat Adleman's experiment [2], the tools of molecular biology have been designed and implemented for a narrow range of experiments. As such, they may have unconsidered side effects when used for computations. For example, errors in the ligation process, such as uncharacterized products, are typically unimportant to biologists because they incorporate the ligation products into cells that not only filter out these incorrectly ligated products, but also amplify the correct ones. In short, it is difficult to predict what will happen when standard molecular biology protocols are applied to DNA computing.

Although the main double encoding hypothesis could not be tested, some important observations could be nonetheless made.

First, it can be difficult to adapt a single encoding into a double encoding. For example, when adapting the encoding used in this paper for single encoding to double encoding, the ability to amplify any given bit was lost. Furthermore, two additional oligos, M_0 and M_1, needed to be added to the encoding to be able amplify the key. Not only did this make the encoding less elegant, it increased the cost of the experiment.

Second, double encoding is more complex. Encoding all keys using single encoding takes $O(1)$ time. Double encoding requires $O(n^2)$ time if each key is created separately. This is a significant drawback of double encoding, as it cripples the parallelism of the DNA computer. It should be noted that an $O(n)$ method was suggested by Boneh *et al* [6]. However, the $O(n)$ technique has its own share of problems since it requires the use of bead separation, the same error-prone experiment that it seeks to improve!

Third, double encoding noticeably increases the length of the resulting DNA strands. For instance, consider a single encoding where the resulting DNA strands are n-bp long. If the entire strand is double encoded, the resulting strand will be $2n$-bp long. While not a problem for molecular programs with short bit strings, molecular programs with very long bit strings, such as the DES-breaking program proposed in [5], would be highly affected. This is due to the fact that as DNA strands become longer, they are more prone to breaking and forming undesired hairpins.

Finally, double encoding may not actually be addressing the problem with separation by hybridization. If a magnetic probe is missing a desired DNA strand, it may be because it never comes near it in the solution, and not because it could not find a suitable binding location.

4 Summary and Conclusion

4.1 Summary

In this paper two topics were investigated: molecular codebreaking and double encoding. Since neither of these two technologies had been carried out in the laboratory, their feasibility in real-world implementations was evaluated.

Molecular codebreaking is a DNA computing technique that uses DNA and molecular biology experiments to recover a key used to map a plaintext to a ciphertext. Since a molecular codebreaker had never been constructed, the molecular biology techniques that make up a molecular codebreaker were evaluated. This was done by designing a stripped-down proof-of-concept version of a DES-breaking molecular codebreaker [5]. The proof-of-concept machine was called MCB-1. Several of the molecular biology experiments that make up MCB-1 were examined. They were ligation, gel electrophoresis, PCR, graduated PCR and bead separation. Each of these techniques, save bead separation, was successfully carried out in the laboratory, commented upon, and found to be feasible with certain reservations. In the case of bead separation, preliminary experiments were carried out with promising initial results.

Double encoding is a proposed [6] error resistance technique that intends to reduce the number of false negatives that occur during bead separation. This paper intended to answer the question of whether or not double encoding was superior than simply doubling the amount of DNA in an experiment. Therefore, a double encoding implementation was devised, and an attempt to carry it out in the laboratory was undertaken. However, double encoding was not successfully implemented. The failure is attributed to the unpredictable outcomes that arise when applying specialized molecular biology techniques to situations for which they were never intended. Several issues associated with practically implementing double encoding were identified. These issues were the difficulty in adapting single encodings to double encodings, the enormous performance penalty associated with generating double encoded strands, the limitations imposed by strand length increases, and the possibility that double encoding may not reduce the number of false negatives.

4.2 Future Work

4.2.1 Double Encoding

Double encoding did not yield successful results. It is suggested that a ligation-based double encoding is impractical in a DNA computer. However, if the double encoded strands could be made by an alternative method, the hypothesis that double encoded strands are better than doubled DNA strands could still be tested. Some alternative methods for acheiving double encoding are:

- Using a DNA synthesizer to create the double encoded strands by employing a mix-and-split combinatorial synthesis technique, as was done in [8] to create variable assignments.
- Purchasing pre-made double encoded strands (for small problems).

As mentioned, double encoding need not be used exclusively with MCB-1. If it appears useful in future experiments it could also be adapted for use with Adleman's machine or any other suitable DNA computer.

4.2.2 Molecular Codebreaking

The molecular codebreaking program MCB-1 was not fully implemented. Thus, it is highly recommended that MCB-1 be carried out as future work. In particular the following portions of the program were not implemented:

- GPCR tests on keyligos 00, 10 and 11. GPCR was only tested for the 01 keyligo.
- Using bead separation to separate based on sequence. Only preliminary work was done using bead separation to separate DNA double-strands into single-strands.
- The Append operation. This operation requires the use of bead separation and ligation and is an important part of MCB-1.

Once MCB-1 has been fully demonstrated, it is recommended that longer bit strings be tested, as well as more complicated encryption schemes.

4.3 Conclusion

The objective of our research was to evaluate the feasibility of undertaking molecular biology experiments associated with using DNA computing to solve problems not suited for a conventional computer. The experiments performed here showed that implementing double encoding using ligation is difficult and impractical. The experiments furthermore suggested that breaking encryption using a DNA computer is feasible as long as the key and ciphertext remain sufficiently small.

Will we one day use DNA computers in the same manner as we use silicon computers? Probably not in the near future. DNA computing as it stands is a time-consuming and delicate operation. However, it should be noted that silicon computers had similar beginnings. Indeed, although DNA computers do not seem feasible now, the massive parallelism offered by DNA is a property that cannot be ignored by researchers. Moreover, unlike other unconventional computers, such a quantum computers and accelerating machines, DNA computers can be implemented now, with technology that exists today. For these and other reasons, the future of DNA computing looks promising, and we may indeed one day see DNA computers used to solve complex problems that are infeasible on silicon computers.

References

[1] R. Adar, Y. Benenson, G. Linshiz, A. Rosner, N. Tishby, and E. Shapiro. Stochastic computing with biomolecular automata. *Proceedings of the National Academy of Science of the USA*, 101(27), 2004.

[2] L. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(5187), 1994.

[3] Y. Benenson, B. Gil, U. Ben-Dor, R. Adar, and E. Shapiro. An autonomous molecular computer for logical control of gene expression. *Nature*, 429, 2004.

[4] Y. Benenson, T. Paz-Elizur, R. Adar, E. Keinan, Z. Livneh, and E. Shapiro. Programmable and autonomous computing machine made of biomolecules. *Nature*, 414, 2001.

[5] D. Boneh, C. Dunworth, and R. Lipton. Breaking DES using a molecular computer. In E. Baum and R. Lipton, editors, *DNA based computers*, volume 27 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, pages 37–66. American Mathematical Society, 1995.

[6] D. Boneh, C. Dunworth, R. Lipton, and J. Sgall. Making DNA computers error resistant. In L. Landweber and E. Baum, editors, *DNA Based Computers II*, volume 44 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1996.

[7] R. Braich, N. Chelyapov, C. Johnson, P. Rothemund, and L. Adleman. Solution of a 20-variable 3-SAT problem on a DNA computer. *Science*, 296, 2002.

[8] R. Braich, C. Johnson, P. Rothemund, D. Hwang, N. Chelyapov, and L. Adleman. Solution of a satisfiability problem on a gel-based DNA computer. In *DNA: International Workshop on DNA-Based Computers*. LNCS, 2000.

[9] Guarnieri, Fliss, and Bancroft. Making DNA add. *Science*, 273, 1996.

[10] P. Kaplan, G. Cecchi, and A. Libchaber. Molecular computation: Adleman's experiment repeated. Technical Report TR-95-120, NEC Research Institute, 1995.

[11] R. Lipton. DNA solution of hard computational problems. *Science*, 268, 1995.

[12] Q. Liu, Z. Guo, A. Condon, R. Corn, M. Lagally, and L. Smith. A surface-based approach to DNA computation. In L. Landweber and E. Baum, editors, *DNA Based Computers II*, volume 44 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1996.

[13] Q. Liu, L. Wang, A. Frutos, A. Condon, R. Corn, and L. Smith. DNA computing on surfaces. *Nature*, 403, 2000.

[14] Q. Ouyang, P. Kaplan, S. Liu, and A. Libchaber. DNA solution of the maximal clique problem. *Science*, 278(5337):446–449, 1997.

[15] S. Roweis, E. Winfree, R. Burgoyne, N. Chelyapov, M. Goodman, P. Rothemund, and L. Adleman. A sticker-based model for DNA computation. *Journal of Computational Biology*, 5(4):615–630, 1998.

[16] K. Sakamoto, H. Gounzu, K. Komiya, D. Kiga, S. Yokoyama, T. Yokomori, and M. Hagiya. Molecular computation by DNA hairpin formation. *Science*, 288(5469):1223–1226, 2000.

[17] X. Su and L. Smith. Demonstration of a universal surface DNA computer. *Nucleic Acids Research*, 32(10):3115–3123, 2004.