5-2015

A forensics software toolkit for DNA steganalysis.

Marc Bjoern Beck
University of Louisville

A FORENSICS SOFTWARE TOOLKIT FOR DNA STEGANALYSIS

By

Marc Bjoern Beck
B.S., Computer Science, Brescia University, 2007
M.S., Industrial Technology, Morehead State University, 2009

A Dissertation
Submitted to the Faculty of the
J. B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy
in Computer Science and Engineering

Department of Computer Engineering and Computer Science

University of Louisville

Louisville, Kentucky

May 2015

A FORENSICS SOFTWARE TOOLKIT FOR DNA STEGANALYSIS

By

Marc Bjoern Beck
B.S., Computer Science, Brescia University, 2007
M.S., Industrial Technology, Morehead State University, 2009

A Dissertation Approved On

April 20, 2015

By the following Dissertation Committee:

_____
Roman V. Yampolskiy, Ph.D., Dissertation Director

_____
Ahmed H. Desoky, Ph.D., Co-Advisor

_____
Eric C. Rouchka, Ph.D.

_____
Ibrahim N. Imam, Ph.D.

_____

John F. Naber, Ph.D.

ACKNOWLEDGEMENTS

I would like to express my special appreciation and thanks to my advisors Dr. Yampolskiy and Dr. Desoky and the other members of my committee Dr. Rouchka, Dr. Ouyang, and Dr. Naber for their extreme patience in the face of numerous obstacles. I also would like to thank Dr. Imam, who kindly agreed to serve as fifth committee member after Dr. Ouyang left the department.

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of this dissertation project. I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

I would also like to mention Chris Card and Sam Hasinoff from the online community who answered questions regarding cryptography software they developed as well as Patrick McClure, whose project provided some of the building blocks for some of my algorithms, my friend and colleague Ana Stanescu at Kansas State University, and also all my friends who helped me with the debugging of source code. I am also thankful to Dr. Monica Rodriguez, who provided me with data that helped determine the frequency of occurrence of letters, words, and letter combinations in the English language.

Last but not least I want to thank Faezeh Taffazoli for helping me format this document.

ABSTRACT

A FORENSICS SOFTWARE TOOLKIT FOR DNA STEGANALYSIS

Marc B. Beck

April 20, 2015

Recent advances in genetic engineering have allowed the insertion of artificial DNA strands into the living cells of organisms. Several methods have been developed to insert information into a DNA sequence for the purpose of data storage, watermarking, or communication of secret messages. The ability to detect, extract, and decode messages from DNA is important for forensic data collection and for data security. We have developed a software toolkit that is able to detect the presence of a hidden message within a DNA sequence, extract that message, and then decode it. The toolkit is able to detect, extract, and decode messages that have been encoded with a variety of different coding schemes. The goal of this project is to enable our software toolkit to determine with which coding scheme a message has been encoded in DNA and then to decode it. The software package is able to decode messages that have been encoded with every variation of most of the coding schemes described in this document. The software toolkit has two different options for decoding that can be selected by the user. The first is a frequency analysis approach that is very commonly used in cryptanalysis. This approach

is very fast, but is unable to decode messages shorter than 200 words accurately. The second option is using a Genetic Algorithm (GA) in combination with a Wisdom of Artificial Crowds (WoAC) technique. This approach is very time consuming, but can decode shorter messages with much higher accuracy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This project is part of an emerging new area of study called DNA Steganography and combines elements from various different areas such as forensics, cryptography, bioinformatics, language processing, and artificial intelligence. Progress in bioinformatics and molecular biology in combination with already established methods and algorithms from the field of cryptography has led to the development of DNA Steganography with a number of papers having been published in the last two decades [1-13].

Bioinformatics in itself is already an interdisciplinary field of science, combining computer science, statistics, mathematics, and engineering to study and process biological data. The goal of bioinformatics is to develop methods and software tools for understanding biological data, especially the study of DNA sequences. Over the past few decades rapid developments in genomic and other molecular research technologies and developments in information technologies have combined to produce a tremendous amount of information related to molecular biology.

DNA sequences that are processed by our software are downloaded from online databases such as GenBank in the form of FASTA files. The FASTA file format is a

common file standard in bioinformatics and originates from the FASTA software package. The name stands for Fast-All, because it works with all alphabets, while previous formats were limited to protein (FAST-P) or nucleotide (FAST-N) alignment. This format is text-based and is used to represent nucleotide sequences or peptide sequences in single-letter codes. These sequences are usually preceded by a header containing sequence names and comments. The header usually starts with a ">" (greater-than) symbol and comprises the first line of the file. Each subsequent line has usually a maximum of 70 to 80 characters. Our software has been designed to accommodate these properties of FASTA files, but is also able to process *.txt files as well.

Deoxyribose Nucleic Acid (DNA) is the carrier of hereditary information for every living organism. DNA is a double helix with two anti-parallel strands containing four different nucleotides, which are distinguished by one of the four bases adenine (A), cytosine (C), guanine (G), and thiamine (T) [14]. The two strands form base pairs of interacting complementary bases (A-T and C-G) held together by hydrogen bonds. DNA has the potential to store vast amounts of data using combinations of those four nucleotides within genomes that can range to several billion bases in length [15].

Contained within genomic sequences are regions that code for genes that produce proteins which are collections of amino acids. In the process of translation, an mRNA sequence that has been transcribed, or copied, from the gene coding region is used as a template to transform from the four base code of DNA to the 20 base code of amino acids. The process by which this occurs, known as the genetic code, was first uncovered by Marshall Nirenberg [16]. In gene coding regions, a codon refers to a sequence of three nucleotides that determines which amino acid will be added next during protein

synthesis. With four nucleotides, this allows 4^3=64 possible combinations. A codon refers to a sequence in a gene coding region of three nucleotides that determines which amino acid will be produced next during protein synthesis. Each codon encodes for one of 20 amino acids, with exception of the three STOP codons TAA, TAG, and TGA [17], thus allowing for degeneracy where multiple codon sequences code for the same amino acid. In DNA steganography, characters of messages may be encoded by variable lengths of DNA sequences that may or may not be three bases in length. For the purpose of this manuscript, we will refer to encoding patterns that encode alphanumeric characters of messages as *codeons* in order to distinguish them from codons and avoid confusion.

The ability to manipulate DNA and to create artificial DNA sequences allows the insertion of artificial messages into those sequences. In the past, messages that have been inserted into DNA could only be retrieved if the coding scheme with which the message has been encoded was known. In this dissertation we describe the development of a software package that is capable of detecting the presence of a message within a DNA sequence, determining how the message has been encoded based on statistical analysis, and then decoding that message.

With DNA sequencers becoming faster, more efficient, more compact, and more affordable [18], and with recent advances in creating artificial DNA, there is an increasing possibility that this kind of technology can be abused for espionage or criminal purposes in the near future. Therefore it is necessary to develop methods to counter attempts to hide information in the DNA of living organisms.

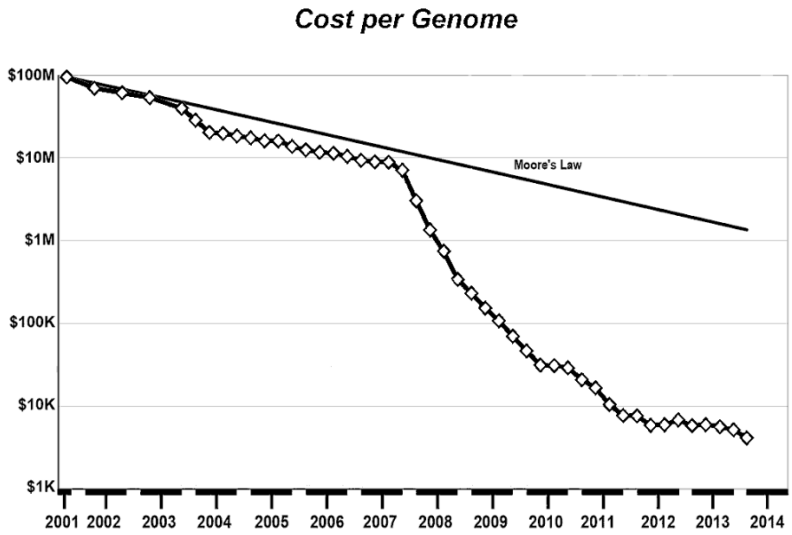


Figure 1.1: Cost for sequencing a genome (modified from[19])

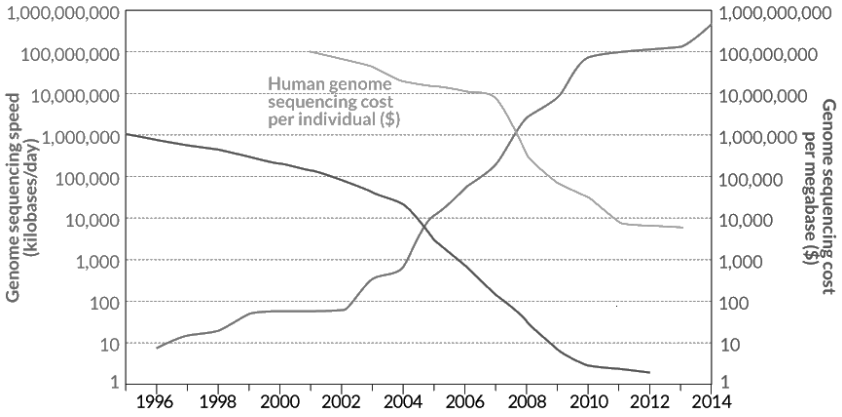


Figure 1.2: Speed of sequencing a genome [20]

It is difficult to generate a graph that shows how the amount of data that can be encoded in DNA has increased over the past 15 years, since the information is not consistent. Some researchers did not provide the length of the encoded message, others provided the number of words instead the number of bytes or characters. Also, some researchers did not try to encode as many characters as it was possible to encode at the time.

CHAPTER 2

RELATED WORK

2.1. DNA Computing

DNA computing is an emerging new research field that uses DNA molecules instead of traditional silicon-based microchips. The first researcher to demonstrate the computing capability of DNA was Leonard Adelman of the University of Southern California, who in 1994 developed a method of using DNA for solving an instance of the directed Hamiltonian path problem [21].

In 1997, Ogihara and Ray demonstrated that DNA computers can simulate Boolean AND and OR gates [22]. The advantage of DNA computers is that they are smaller and faster than traditional silicon computers, and that they can be easily used for parallel processing. DNA has also been used as a tool for cryptography and cryptanalysis, using molecular techniques for its manipulation [17].

In 2004, Benenson et.al [23] developed an autonomous, molecular scale computer, which uses biological molecules as input data and biologically active molecules as outputs. This computer logically analyzes the levels of messenger RNA species, and in response produces a molecule capable of affecting levels of gene expression [23].

Bogard and Rouchka describe how multiple sequence alignment can be used for error reduction in DNA computing [24].

As described by Amir et al. [25], the nanoscale folding of DNA, which is called DNA origami, can be utilized for the fabrication of nanoscale architectures that behaved as logical gates (AND, OR, XOR, NAND, NOT, CNOT, and a half adder).

DNA as Storage Medium

In 2013, the total amount of digital information worldwide was 4.4 zettabytes and is predicted to reach 44 zettabytes by 2020 [26]. Shrivasti et al. [27] describe the shortcomings of silicon and other materials used to manufacture data storage media. These shortcomings include limited, non-renewable resources, relatively low storage density and relatively low access rates.

DNA is being investigated by a number of independent researchers as an ultra-compact, long-term data storage medium and a stegomedium for hiding messages. Just like binary code, DNA is a coding medium. DNA strands contain information that can be interpreted and copied, just like the sequences of ones and zeros on a hard drive or in RAM [2].

Instead of expressing a message as a series of ones and zeros, it is expressed in DNA code as a series of As, Cs, Gs, and Ts, representing the four nucleotides. These nucleotides can easily be used for the encoding of binary information. The easiest way to hide a secret message in a binary sequence is to add the message and increase the overall size of the sequence. This however would make the message easier to detect and may change the functionality of the sequence. That means in order to embed a message, one

should not arbitrarily append or intersperse information. A complete understanding of the original message and the machinery that processes it is necessary in order to be able to modify some portion in an intelligent manner such that the data is not functionally or perceptibly altered. In the same manner, a sequence of nucleotides should not be changed blindly just to insert a secret message [2].

A number of algorithms have been developed to encode a message in DNA characters and either disguise these messages as novel DNA sequences or encapsulate them within existing ones. It has been proven that it is possible to insert artificial DNA components that contain encoded information into the genomes of living organisms [5-7, 9, 17, 28-32].

Craig Venter, who led the private effort to sequence the human genome, managed to create the first cell with a synthetic genome in 2010. The J. Craig Venter Institute (JCVI) took a computer file containing the DNA sequence of the bacterium *Mycoplasma mycoides*, modified it, produced physical DNA from this sequence, and inserted this DNA into a cell, which then reproduced under control of the new DNA to create a new bacterium. This project encoded 7920 bits [9].

Using DNA as storage medium has many advantages, such as long life, redundancy, and high density. According to Bancroft et al. [33] about 200 novels or other data each equivalent in size to "A Tale of Two Cities" could be stored in a DNA microchip with the area of a postage stamp.

Yachie et al. [5] demonstrated the possibility to use DNA of living organisms as a data storage medium by inserting the message "E=mc^2 1905!" into the genome of *B.*

subtilis. Multiple copies were created and over 99% of the encoded data was later recovered using sequence alignment methods.

Living organisms are a great storage medium when it comes to preserving data over timespans ranging in millions of years. When an organism reproduces, it automatically creates a copy of the data contained in its DNA. In addition, selective pressure and DNA error correction reduce the risk of the data being destroyed by random mutations. It has been suggested to use cockroaches, which are known for their resilience and high reproduction rate, as living time capsules for storing every issue of The New York Times Magazine for a certain year in their DNA which could theoretically be retrieved 1000 years later [34].

Type	Life Expectancy	Capacity
DNA	Millions of years (over generations) 521 years (dead organism)	455 Exabytes per gram[27]
Hard disk	~10 years	Up to 20TB (2015)
SSD	~10 years	Up to 4 TB (2015)
CD	10 to 100 years [35]	800 MB
DVD	20 to 100 years [35]	Up to 17GB
Blue Ray	30 to 100 years	25GB (single layer) 128 GB (BDXL)
USB flash drive	~10 years, depending on usage	Up to 1TB (2015)
Tape	~30 years	Up to 35 TB

Table 2.1: Life expectancy and storage capacity of various data storage media compared to DNA.

As shown in table 2.1, 1 gram of DNA has a storage potential of 455 exabytes of information. Conventional media would require roughly 2 million times that volume for the same amount of information. Researchers at the Swiss Federal Institute of Technology in Zurich encoded the Swiss federal charter from 1291 and the Archimedes Palimpset totaling 83 kb of data in DNA and kept the DNA versions at temperatures

between 60 and 70 degrees Celsius (140-158 degrees Fahrenheit) to simulate ageing. Both documents remained readable without error [36]. The same group of researchers also encapsulated DNA in microscopic spheres of glass in order to mimic the way fossils keep DNA intact. They did not mention any research regarding the effect of the exposure to ultraviolet light over time.

In 2014 Dr. Ido Bachelet [13] mentioned at the Geektime 2014 conference that he was working on a project to insert Wikipedia into the DNA of an apple, but was not specific if he was just inserting the text of the articles, or if he was including revisions, images, and other data. The rate of compression, if any, was not mentioned, either. The amount of data could range anywhere between 8.8 GB and 5TB. Bachelet also mentioned a project inserting data encoding the famous painting of the Mona Lisa into Mouse DNA [13]. Both projects are ongoing as of April 2015 and there are no publications yet.

2.2. Error Correcting Approaches

Even though mutations are rare, occurring at a rate between 10^{-11} and 10^{-7} per base per replication in bacteria and higher eukaryotes [37], it is necessary to consider some form of error detection and error correction since a mutation can destroy the encrypted message in the DNA sequence. According to Yachie et al. [5], inserting the data redundantly into multiple loci of the genome is sufficient to allow the retrieval of stable and compact data without the need for template DNA, parity checks, or error-correcting algorithms.

The comma code and the alternating code provide a form of error detection capability by encoding the message in a distinguishable pattern [38]. Arita [29] developed a

comma-free code that has error correction capabilities. The message is translated into binary as an intermediary step. A parity bit is used in the binary code to keep the respective number of ones and zeroes odd.

The DNA-Crypt software developed by Heider and Barnekow [6] also translates messages into binary before encoding it in DNA code. It uses a very thorough approach to error detection by employing two error correction codes: the 8/4 Hamming-code and the WDH-code. The Hamming code is an error detection and correction code invented by Richard Hamming in 1950 [39] and the WDH code was developed by Andrew S. Tanenbaum and is used in computer networks [40]. The 8/4 Hamming code is more compact, but it can correct fewer errors than the WDH code. DNA-Crypt has an integrated fuzzy controller using Singleton-fuzzyfication. The fuzzy controller decides which of the two error detecting codes should be used, or none at all. This decision is based on the individual mutation rate of the DNA sequence that contains the secret message, the length of the sequence, and its stability over time. An answer is determined from those three factors by a set of rules based on heuristics [6].

A team at the Swiss Federal Institute of Technology in Zurich that is researching DNA as a data storage medium uses a Reed-Solomon code [36] for error correction. Reed-Salomon codes are very commonly used in data storage and data transmission applications. This includes the use for correcting burst errors in CDs, DVD's and Blu-ray discs caused by media defects, and the use in satellite and deep space applications.

2.3. DNA Cryptography

DNA cryptography is a relatively new area of research. Soni, Soni, and Mathariya [41] describe approaches that use DNA sequences from public databases as key for one-time pad (OTP) algorithms, as well as a method based on the DNA splicing technique.

When working with one-time pad algorithms, the plaintext is combined with a secret random key or pad which is used only once. This is done by using an XOR operation, a typical modular addition or a similar technique.

Also, Soni et al. [41] describe an algorithm which avoids the usage of both purely mathematical symmetric and asymmetric algorithms by making use of asymmetric cryptographic principles and an advanced asymmetric algorithm based on DNA.

2.4. Hiding Data in DNA

Steganography is the science of hiding information by transmitting secret messages through unsuspicious cover carriers in a way that makes the presence of any embedded messages undetectable to a third party. The term steganography originates in the Greek language and means, "covered writing". While the goal of cryptography is to make a message unreadable, steganography aims at avoiding suspicion to the existence of a hidden message [3]. Due to its properties as a data storage medium, DNA can be used for steganography (stegomedium).

Since DNA encoded information can be copied just like digital information, there is a high possibility of theft of intellectual property. Therefore, it would be wise to follow the

fundamental requirements and principles of hiding information in digital data, which are the following [42] :

• be reasonably easy to be placed and detected by the legitimate party

• be difficult for attackers to detect and erase

• be credible in case of a dispute

• be robust against compression, filtering, or truncation

• avoid unnecessary overhead

• not significantly change the meaning or function of the original data

• have error detection/correction and/or redundancy for the data being hidden

It is not easy to filter DNA for the purpose of defeating a watermark, unlike other cover media such as audio or video. It is possible to modify audio and video information in such a way that significant degradation of the signal is not qualitatively noticeable. One important fact is that it is difficult to modify DNA without a sufficient understanding of the sequence, especially if it codes for specific biological functions. This would force an attacker to do a substantial amount of original work. That means the requirement for robustness against compression, filtering, or truncation does not apply here. A DNA and RNA data hiding technique should however adhere to the other basic criteria [2].

Recent developments make the use of DNA as a stegomedium for concealing, storing, and transmitting messages more feasible. This means that there will be an increasing need for forensic methods to extract and decode such messages in the near future. Not very many such methods are in existence so far.

It is possible to insert not only text, but also images and many other forms of digitizable data into a DNA sequence. DNA could also be used by criminal organizations

to hide illegal information. For those reasons, it is becoming important to develop forensic tools that can detect, extract, and decode information that has been hidden in DNA.

Table 2.2: Genetic code for protein translation (codons that code for the same amino acid regardless the third position are highlighted) [17, 43].

First Position		Second Position of codon				Third Position
		T	C	A	G	
T	T	TTT [F]	TCT [S]	TAT Tyr [Y]	TGT [C]	T
		TTC [F]	TCC [S]	TAC Tyr [Y]	TGC [C]	C
		TTA [L]	TCA [S]	TAA [end]	TGA [end]	A
		TTG [L]	TCG [S]	TAG [end]	TGG [W]	G
	C	CTT [L]	CCT [P]	CAT His [H]	CGT [R]	T
		CTC [L]	CCC [P]	CAC His [H]	CGC [R]	C
		CTA [L]	CCA [P]	CAA Gln [Q]	CGA [R]	A
		CTG [L]	CCG [P]	CAG Gln [Q]	CGG [R]	G
	A	ATT [I]	ACT [T]	AAT Asn [N]	AGT [S]	T
		ATC [I]	ACC [T]	AAC Asn [N]	AGC [S]	C
		ATA [I]	ACA [T]	AAA Lys [K]	AGA [R]	A
		ATG [M]	ACG [T]	AAG Lys [K]	AGG [R]	G
	G	GTT [V]	GCT [A]	GAT Asp [D]	GGT [G]	T
		GTC [V]	GCC [A]	GAC Asp [D]	GGC [G]	C
		GTA [V]	GCA [A]	GAA Glu [E]	GGA [G]	A
		GTG [V]	GCG [A]	GAG Glu [E]	GGG [G]	G

An obvious choice of a location for inserting a message into a genome would be a noncoding genomic region. However, those regions might be involved in different regulations which are as of yet unknown [28]. Inserting data there might possibly kill the organism. Therefore Arita et al. [28] suggested that it may be a more reliable solution to encode the message in the protein coding regions of genes. There are 20 amino acids and one stop symbol using a total of 64 possible codons [28]. Two or more codons often code for the same amino acid. This means that many codons are redundant and it is possible to use this redundancy to encode additional information. Many of these redundant, or synonymous, codons typically differ in their third position, also known as the wobble

base [3]. In Table 2.2, codons that encode for the same amino acid regardless which base occupies their third position are highlighted. These are the codons that are used to embed messages.

Table 2.3: Research on data hiding in DNA

Researcher	Year	Coding	Message	Location	Organism
Clelland et al.[1]	1999	Substitution	June 6 invasion: Normandy	Artificial	Human
Brenner et al.[31]	1999	Comma code	Not reported	Bsp120I	E.coli
Shimanovsky et al.[2]	2002	BinaryToRNA	0100100010010 0010101100100 1101001101110 1	theoretical	Theoretical
Wong et al. [3]	2003	Substitution	Not reported	Not reported	Deinococcus radiodurans
Arita and Ohashi [28]	2004	Arita	“AO2KEIO1-F”	ftsZ gene	B. subtilis RIK8
Tanaka et al.[4]	2005	Substitution	“MESSAGE”	Artificial sequence	Artificial DNA strand
Yachie et al. [5]	2007	Keyboard scan	“E=mc^2 1905!”	metB and proB	B.subtilis BEST2136
Heider and Barnekow [6]	2007	DNA-Crypt	“TB”	Vam7 sequence	Saccharomyces cerevisiae CG783
Jiao and Gouette [7]	2009	ASCII 8 bit binary	“CODING”	tatAD gene	B. subtilis
Ailenberg and Rotstein[8]	2009	Improved Huffman	Text:Lyrics “Mary had a little lamb”	SacI/KpnI	PBluescript based plasmid
Ailenberg and Rotstein[8]	2009	Improved Huffman	Music: Tune “Mary had a little lamb”	SacI/KpnI	PBluescript based Plasmid
Ailenberg and Rotstein[8]	2009	Improved Huffman	Image: lamb	SacI/KpnI	PBluescript based plasmid
Venter et	2010	Substituti	Multiple	Not	Artificial

al.[9]		on	messages	reported	bacterium
Mousa et al.[10]	2011	Contrast mapping	random numbers	RSNn256728	Random sequence of nucleotides
Church et al. [11]	2012	Binary to DNA	"Regenesis" (book with 53,000 words)	Artificial	Theoretical
Goldman et al. [12]	2013	Substitution	Misc. data 757,051 bytes total	Artificial	Theoretical
Bachelet [13]	2014	Binary	Mona Lisa	Not reported	Mouse
Bachelet [13]	2014	Binary	Entire content of Wikipedia	Not reported	Apple

2.5. Coding Schemes for Hiding Data in DNA

A code is an algorithm which uniquely represents symbols from some source alphabet, by symbols or strings of symbols in a target alphabet. In our case, the source alphabet is the English alphabet plus digits and punctuation characters, and the target alphabet consists of the four nucleotides. A coding scheme is a set of rules that determines which symbol of the source alphabet is represented by which symbol in the target alphabet. A variety of different coding schemes has been developed to encode alphanumeric characters in DNA sequences.

2.5.1. Types of Coding Schemes

The coding schemes for inserting messages into DNA that have been developed can be grouped into three categories: schemes using direct translation, schemes that use intermediate steps for error detection, and schemes that have been optimized for detectability or efficiency.

The first category uses a straightforward approach by substituting a sequence of nucleotides of length n for each alphanumeric symbol [10,12]. Since the codeon in this case is of length n, up to 4n distinct characters can be encoded. Given the codeon length of n, there are 4n! possible coding schemes. The coding schemes developed by Clelland [30] and Wong [3] fall into this category.

The second category of coding schemes consists of more complex schemes that use several intermediate steps, such as translating a message into binary before using a coding table to translate it into nucleotides. This is often done for error detection, since there are many proven error detection algorithms for binary messages.

The third category of coding schemes consists of schemes that were designed to meet certain criteria, such as providing error detection capability, being economical, or being easy to detect. The comma code, the alternating code, and a coding scheme based on the Huffman code [18] fall into this category.

2.5.2. Clelland's Coding Scheme and Wong's Coding Scheme

The coding scheme developed by Clelland et al. [30] is very similar to the one developed by Wong et al. [3]. They are both extensions of the 3-base codon encoding used by the genetic code. Since there are 4^3=64 possible distinct characters that can be encoded, this scheme allows for all 26 characters of the English alphabet, the digits 0-9, and special characters. Both coding schemes do not use all possible codons.

Table 2.4: Coding table for Clelland's coding scheme.

Character	DNA		Letter	DNA
A	CGA		V	CCT
B	CCA		W	CCG
C	GTT		X	CTA
D	TTG		Y	AAA
E	GGT		Z	AAT
F	ACT		0	TTA
G	TTT		1	ACC
H	CGC		2	TAG
I	ATG		3	GCA
J	AGT		4	GAG
K	AAG		5	AGA
L	TGC		6	GGG
M	TCC		7	ACA
N	TCT		8	AGG
O	GGC		9	GCG
P	GGA		SPACE	ATA
Q	AAC		,	TCG
R	TCA		.	GAT
S	ACG		:	GCT
T	TTC		;	ATT
U	CTG		-	ATC

Table 2.5: Coding table for Wong's coding scheme.

Character	**DNA**		Character	DNA
A	AGG		0	AAA
B	AGT		1	AAC
C	ATA		2	AAG
D	ATC		3	AAT
E	ATG		4	ACA
F	ATT		5	ACC
G	CAA		6	ACG
H	CAC		7	ACT
I	CAG		8	AGA
J	CAT		9	AGC
K	CCA		SPACE	GCA
L	CCC		,	GCG
M	CCG		.	GGA
N	CCT		:	GCC
O	CGA		;	TAC
P	CGC		-	GCT
Q	CGG		!	GGC
R	CGT		(	GGG
S	CTA		)	GGT
T	CTC		`	GTA
U	CTG		‘	GTC
V	CTT		“	GCC
W	GAA		?	TAA
X	GAC		/	TAG
Y	GAG		[	TAT
Z	GAT		]	TCA

2.5.3. DNA-Crypt

The DNA-Crypt coding scheme developed by Heider and Barnekow [10] translates a message into a five bit sequence, where one bit serves as parity bit to keep the respective number of ones and zeros odd. The other four bits are translated into nucleotides, with

two bits per nucleotide. As mentioned earlier, it employs two error correction codes, the 8/4 Hamming-code and the WDH-code.

Table 2.6: Coding table for the DNA-Crypt coding scheme, with 00=T; 01=G; 10=C; 11=A and the last bit being used as a parity bit.

Letter	Binary		Letter	Binary
A	00000		N	01101
B	00001		O	01110
C	00010		P	01111
D	00011		Q	10000
E	00100		R	10001
F	00101		S	10010
G	00110		T	10011
H	00111		U	10100
I	01000		V	10101
J	01001		W	10110
K	01010		X	10111
L	01011		Y	11000
M	01100		Z	11001

2.5.4. ASCII Coding Scheme

Another coding scheme implements the algorithm described by Jiao and Gouette [17] which inserts a message into the noncoding region of an existing DNA sequence. This method consists of several steps:

1) Convert each character in the message into its ASCII representation.
2) Convert the ASCII code from decimal into binary.
3) Converting binary to DNA by replacing 00 with A, 01 with C, 10 with G, and 11 with T.
4) Insert message into a carrier DNA sequence.

Steps 1-3 are referred to as the ASCII coding scheme throughout the remainder of this dissertation. The fourth step can be applied to other coding schemes if the message is to be inserted into a coding DNA region. This is done by replacing the last bits of redundant codons in the carrier sequence with characters from the message sequence. The ASCII coding scheme makes it possible to encode uppercase letters, lowercase letters, numbers, and special characters. Each character is represented by a sequence of four bases.

2.5.5. Yachie's Coding Scheme

Yachie et al. [5] developed a coding scheme that is very similar to ASCII encoding. Instead of ASCII it uses the keyboard scan code for each character. The keyboard scan code, which is hexadecimal, is converted into binary, and then translated into DNA using the coding table below.

Table 2.7: Coding scheme used by Yachie et al.

Message	**E=mc^_1905**		**Encryption Key**			
Keyboard Scan Code	%12%24%12%4E%3A %21%55%1E%29%16 %46%45%2E%12%16		AA	0000	AG	1000
			CA	0001	CG	1001
			GA	0010	GG	1010
Hexadecimal Code	1 2 2 4 1 2 4 E 3 A 2 1 5 5 1 E 2 9 1 6 4 6 4 5 2 E 1 2 1 6		TA	0011	TG	1011
			AC	0100	AT	1100
Binary Code	0001 0010 0010 0100 0001 0010 0100 1110 0011 1010 0010 0001 0101 0101 0001 1110 0010 1001 0001 0110 0100 0110 0100 0101 0010 1110 0001 0010 0001 0110		CC	0101	CT	1101
			GC	0110	GT	1110
			TC	0111	TT	1111

2.5.6. Arita's Coding Scheme

Arita and Ohashi [28] translated each letter of the English alphabet as well as an empty space and the characters ‘‘’, ‘.’, ‘&’ into a 6-bit binary sequence. One of the bits serves

as parity bit by keeping both the number of 0s as well as the number of 1s odd for error detection. When a message encoded with this coding scheme is inserted into a coding region of a DNA sequence, a 0 indicates to leave the 3rd base of a codon unchanged, while a 1 indicates that it needs to be changed. In order to extract the encoded message, one needs to compare the sequence that contains the message with the original, unchanged sequence to determine if a base was changed or not [6].

Table 2.8: Coding scheme used by Arita et al.

Letter	Binary		Letter	Binary
A	001000		Q	111000
B	100110		R	010011
C	100101		S	100000
D	001101		T	000100
E	000010		U	101001
F	001110		V	101010
G	010110		W	110100
H	100011		X	110010
I	001011		Y	011100
J	110100		Z	011111
K	101100		Space	000001
L	010101		‘	101111
N	000111		.	110111
O	010000		&	111011
P	110001			

2.5.7. Coding Scheme Based on the Huffman Code

Another coding scheme is based on the Huffman code developed by David A. Huffman [44] and the frequency of letters in the English language from “The Code Book” by Simon Singh [45]. The Huffman code is an entropy encoding algorithm used for lossless data compression. The coding scheme developed by Smith et al. [38] based on this code only encodes letters, but not numbers or special characters. The average codon length in

this case is 2.2 bases. There are 4! possible ways to generate a Huffman code for encoding the 26 letters of the English alphabet, but it is also possible to create a Huffman code-based scheme that includes more characters, such as numbers, punctuation characters, and others.

Table 2.9: Letter frequency in English language and DNA coding scheme using Huffman code [38].

Letter	Freq(%)	DNA		Letter	Freq(%)	DNA
e	12.7	T		w	2.4	AAT
t	9.1	AG		m	2.4	ACA
a	8.2	AT		f	2.2	ACG
o	7.5	GA		y	2.0	ACC
i	7.0	GG		g	2.0	ACT
n	6.7	GC		p	1.9	CCA
s	6.3	GT		b	1.5	CCG
h	6.1	CA		v	1.0	CCT
r	6.0	CG		k	0.8	CCCA
d	4.3	CT		j	0.2	CCCG
l	4.0	AAA		x	0.2	CCCC
c	2.8	AAG		q	0.1	CCCTA
u	2.8	AAC		z	0.1	CCCTG

Ailenberg and Rotstein [8] improved this coding scheme, increasing the number of encoded characters from 26 (the letters of the English alphabet) to 69 (the characters on a computer keyboard). They accomplished this by replacing Cs with As, Gs with Ts and moving CG-rich codons down the frequency table. Furthermore, they divided symbols into three groups using low–base number DNA codons (G, TT, and TA) as group prefixes.

Ailenberg and Rotstein [8] also managed to create Huffman code based coding tables for storing music and for storing images in DNA. By assigning note values and pitches, as well as meters and repeats to DNA codons based on their frequency of occurrence it

was possible to encode the tune of the nursery rhyme “Mary Had a Little Lamb” in DNA. An image was encoded in DNA symbols using a coding table that assigns DNA symbols to shapes and coordinates.

2.5.8. Comma Code

The comma code uses 4-base codons consisting of combinations of A, C, G and T, where G serves as a separator between the different characters. The term comma code may be misleading. It does not mean that G is the encoding for the comma character, but that it separates the encodings for each character. Smith et al. [38] suggest using 5-base codons with a separator every sixth base, but the original paper by Brenner et al. [31] is more descriptive and recommends the use of four bases per codon and a vocabulary made up of eight four-base ‘‘words’’ for biochemical reasons. The gaps between the Gs are filled with either TTAC, AATC, TACT, ATCA, ACAT, TCTA, CTTT, or CAAA. Since this would only allow the encoding of eight characters, combinations of two such words separated by a G are used for each character. This results in a total of 64 possible characters consisting of ten nucleotides each. The comma code encodes lowercase letters, numbers from 0-9, and special characters. The mapping of codons to characters was arbitrarily constructed. A sequence in comma code can easily be identified as containing a message, due to the occurrence of G every five bases, including the beginning and the end of the sequence. The comma code is the least efficient coding algorithm.

Table 2.10: Coding table for the comma code

Character	DNA		Character	DNA
A	GTTACGTTAC		6	GACATGTTAC
B	GTTACGAATC		7	GACATGAATC
C	GTTACGTACT		8	GACATGTACT
D	GTTACGATCA		9	GACATGATCA
E	GTTACGACAT		!	GACATGACAT
F	GTTACGTCTA		?	GACATGTCTA
G	GTTACGCTTT			GACATGCTTT
H	GTTACGCAAA		+	GACATGCAAA
I	GAATCGTTAC		-	GTCTAGTTAC
J	GAATCGAATC		/	GTCTAGAATC
K	GAATCGTACT		*	GTCTAGTACT
L	GAATCGATCA		_	GTCTAGATCA
M	GAATCGTCTA		@	GTCTAGACAT
N	GAATCGACAT		#	GTCTAGTCTA
O	GAATCGCTTT		$	GTCTAGCTTT
P	GAATCGCAAA		%	GTCTAGCAAA
Q	GTACTGTTAC		^	GCTTTGTTAC
R	GTACTGAATC		&	GCTTTGAATC
S	GTACTGTACT		(	GCTTTGTACT
T	GTACTGATCA		)	GCTTTGATCA
U	GTACTGACAT		~	GCTTTGACAT
V	GTACTGTCTA		[	GCTTTGTCTA
W	GTACTGCTTT		]	GCTTTGCTTT
X	GTACTGCAAA		{	GCTTTGCAAA
Y	GATCAGTTAC		}	GCAAAGTTAC
Z	GATCAGAATC		|	GCAAAGAATC
0	GATCAGTACT		<	GCAAAGTACT
1	GATCAGATCA		>	GCAAAGATCA
2	GATCAGACAT		:	GCAAAGACAT
3	GATCAGTCTA		;	GCAAAGTCTA
4	GATCAGCTTT		.	GCAAAGCTTT
5	GATCAGCAAA		,	GCAAAGCAAA

2.5.9. Alternating Code

The alternating code uses 64 codons with six bases per codon, alternating between purines (A or G) at odd positions and pyrimidines (C or T) at even positions. This forms a

pattern that does not occur naturally and can easily be recognized. For the same reason, the bases could be arranged for example in a pattern that has three purines followed by three pyrimidines or vice versa. The alternating code encodes the same characters as the comma code. The decision which codon codes for which character was made arbitrarily [38].

Table 2.11: Coding table for the alternating code.

Character	DNA		Character	DNA
A	ACACAC		6	GTGTGT
B	ACACAT		7	GTGTGC
C	ACATAT		8	GTGCGC
D	ACACGT		9	GTGTAC
E	ACATGT		!	GTGCAC
F	ACATGC		?	GTGCAT
G	ACACGC			GTGTAT
H	ACATAC		+	GTGCGT
I	ACGTAC		-	GTACGT
J	ACGTGT		/	GTACAC
K	ACGTGC		*	GTATGT
L	ACGTAT		_	GTATAC
M	ACGCAC		@	GTACAT
N	ACGCGT		#	GTATAT
O	ACGCGC		$	GTACGT
P	ACGCAT		%	GTACGC
Q	ATGCGT		^	GCGCGC
R	ATGCGC		&	GCGCGT
S	ATGCAC		(	GCGTGT
T	ATGCAT		)	GCGTGC
U	ATGTAT		~	GCGCAT
V	ATGTGT		[	GCGTAT
W	ATGTAC		]	GCGTAC
X	ATGTGC		{	GCGCAC
Y	ATATAT		}	GCACAC
Z	ATATAC		|	GCACAT
0	ATACAC		<	GCATAT
1	ATACAT		>	GCACGT
2	ATATGT		:	GCATGT
3	ATATGC		;	GCATGC
4	ATACGT		.	GCACGC
5	ATACGC		,	GCATAC

2.5.10. Contrast Mapping

Mousa et al. [10] describe a method that formats the DNA sequence that is used to conceal the message by first translating it into binary, and then converting every 6 bits into decimal. Mousa then uses Reversible Contrast Mapping (RCM) to transform a pair of values that represent two consecutive words into another pair of values, with x and y being the first pair of values, and x' and y' being the resulting pair.

$$x'=2x-y;\ y'=2y-x \qquad (2.1)$$

The transformation is restricted to a subdomain in order to prevent overflow and underflow. This subdomain is defined by the equation

$$0 \leq 2x - y \leq L; 0 \leq 2y - x \leq L \qquad (2.2)$$

The values are transformed back by the following formula:

$$x = \left[\frac{2}{3}x' + \frac{1}{3}y'\right];\ \ y = \left[\frac{1}{3}x' + \frac{2}{3}y'\right] \qquad (2.3)$$

The message is inserted according to the flow chart below:

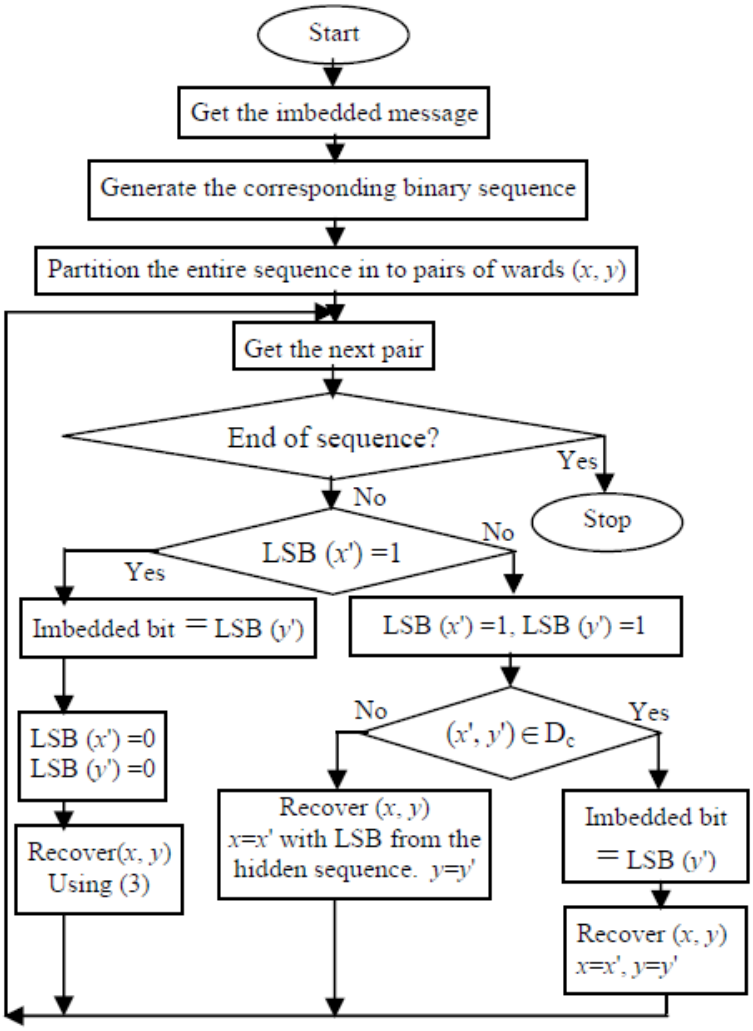


Figure 2.1: Flow chart of the coding algorithm used by Mousa et al. [10].

Mousa's coding scheme can not only be used to conceal messages in DNA sequences, but also for hiding messages in images [10].

2.5.11. Shimanovsky's Coding Scheme

Shimanovsky et al. [2] developed a method to insert a binary message into an mRNA sequence by converting it into a decimal number between one and zero and mapping it to a series of codons using arithmetic encoding and a coding table they devised for that

purpose. Unfortunately we were unable to implement their algorithm and duplicate their results.

2.5.12. Summary of Coding Schemes

The coding schemes differ in codon length, detectability, number of characters that can be encoded, and the number of steps involved in encoding a message. The coding scheme based on the Huffman code is the most economical in terms of codon length, while the comma code is the least economical. The Clelland coding scheme and the Wong coding scheme are the easiest to implement. The comma code, alternating code and DNA-Crypt are the easiest to detect, and DNA-Crypt offers the best error correction.

Inserting a message into a coding region only replaces bases, but does not add new ones. Therefore the size of the genome is only affected if the message is inserted into a non-coding region. The length of the encoded message is the length of the unencoded message multiplied by the codon length of the coding scheme. For example, the message "UNIVERSITY OF LOUISVILLE" is 24 characters long, including spaces. It would be "CTGCCTCAGCTTATGCGTCTACAGCTCGAGCGAATTCCCCGACTGCAGCTAC TTCAGCCCCCCATG", which is 72 characters in Wong's coding scheme and "GTACTGACATGAATCGACATGAATCGTTACGTACTGTCTAGTTACGACATGT ACTGAATCGTACTGTACTGAATCGTTACGTACTGATCAGATCAGTTACGAATC GCTTTGTTACGTCTAGAATCGATCAGAATCGCTTTGTACTGACATGAATCGTT ACGTACTGTACTGTACTGTCTAGAATCGTTACGAATCGATCAGAATCGATCAG TTACGACATG", which is 240 characters in Comma Code.

For inserting pictures, audio, and video files into a DNA sequence it appears to be

efficient to translate the binary representation of the file into DNA code, with each base encoding two bits, for example A=00, C=01, G=10, and T=11.

Table 2.12: Comparison of different coding schemes that use substitution ciphers.

Type	# of chars encoded	# of bases	Developed by
Clelland (basic substitution)	42	3	Clelland et al [46]
Wong(basic substitution)	52	3	Wong et al [3]
Comma	62	10	Smith et al.[38]
Alternating	64	6	Smith et al.[38]
Huffman	26	2-5	Smith et al.[38]
Arita	30	3	Arita and Ohashi[29]
DNA-Crypt	26	5	Heider and Barnekow[6]
ASCII	128	4	Jiao and Gouette [47]

2.6. Encryption and Watermarking of DNA Messages

To make detection even more difficult, it is possible to encrypt a message using modern encryption algorithms such as Data Encryption Standard (DES), RSA, and Number Theory Research Unit (NTRU) before encoding it into DNA.

One application for inserting messages into DNA is watermarking. Three different types of watermarks exist. These are fragile watermarks, semi-fragile watermarks, and robust watermarks. Fragile watermarks are widely used for tamper detection because they fail to be detectable. Semi-fragile watermarks are designed in a way that they can resist benign transformations in order to be able to detect malignant transformations. Robust watermarks are used for copy protection because they can tolerate a designated transformation [48].

Watermarking can help establish brand names for engineered bacteria strains in order to resolve legal disputes regarding gene-related patents [28]. In this application, a researcher creating an artificial DNA sequence can embed a hidden message in this sequence so that ownership of the intellectual property can later be asserted and/or to ensure the integrity of the content. The JCVI, for example, could upload a FASTA file of an artificially created bacteria strain on an online database. If someone else steals the file and claims the work as his or her own, the researchers at JCVI can later prove ownership because only they can recover the watermark.

Watermarking infectious agents can be useful for tracking them back to their source after an accidental release [49]. The usefulness of watermarking for control of agents emerges when this technology is applied by a trusted authorizing entity, which would be in charge of overseeing the distribution of organisms containing unique and confidential watermark sequences to individual research laboratories. These watermarks would distinguish their organisms from those of others in the research community. Laboratories would then only be allowed, or required to use strains that contain their approved watermark. If released, the pathogen in question would be investigated for the presence of an approved watermark. In case such a watermark is found, then information about the possible source could easily be retrieved.

The coding scheme created by Arita and Ohashi [28] and the DNA-Crypt algorithm developed by Heider and Barnekow [6] were both designed for watermarking short trademarks or signatures into genomic DNA.

Researchers at the JCVI inserted four watermarks using a coding scheme with a substitution cipher similar to Clelland's and Wong's into their artificial genome. The first

watermark consists of a copyright-like statement, the coding table for Ventner's coding scheme, and a hidden HTML page. The second, third, and fourth watermarks consist of a list of the authors and three quotations.

Shimanovsky et al [2] suggest different approaches to DNA watermarking, based on the application. They differentiate between chemical DNA, which can be arbitrarily altered without concern and live DNA, which is actually part of a living organism and needs to keep its structural and regulating functions. As an example for chemical DNA, Shimanovsky et al describe a DNA how a computing solution for a traveling salesman problem could be watermarked. In this example, placing the watermark would be accomplished through the selection of an almost-optimal solution instead of an optimal one and hiding it as the watermark inside the destinations path. Here, the watermark is in the solution to a problem and only by knowing an answer that is at least as good someone would be able to remove it. The robustness of this kind of watermark comes from the difficulty of solving the problem.

On the other hand, this kind of technique would not be the best solution for the watermarking of live DNA, where the purpose of a watermark is to protect intellectual property. Genetic coding regions have notable characteristics such as START and STOP codons surrounding them, which would be easily recognizable as a demarcation by an attacker. This information enables an attacker to easily isolate the important sections from a small DNA sequence and start searching for the watermark. What would be even easier than that is replacing the active segment with a neutral sequence that would be identical to the original otherwise. This would effectively remove the watermark [2].

2.7. Finding Data in DNA

Steganalysis is the process of discovering hidden messages [50]. There are two main categories of steganalytic methods: blind steganalysis and specific steganalysis. Blind steganalysis can be used to detect a variety of different steganographic algorithms, even previously unknown ones. The goal of specific steganalysis is to detect a specific known steganographic algorithm by exploring how this particular algorithm works and how it changes the statistics of the cover media [51].

The research on steganalysis is important for several reasons: First, detecting the presence of secret messages can help intercept communication between members of terrorist organizations or other illegal groups. Second, improvements in steganalysis also help to develop better methods for information hiding. Third, better statistical methods for multimedia contents can emerge as a byproduct of steganalysis research. These can then be applied in other related research fields, such as digital forensics [51], or bioinformatics.

Most existing steganalysis approaches focus on images as a stegomedium, especially JPEG images. Audio and video files are also used fairly often. Text documents are not used as often as a stegomedium because they can only hold a smaller amount of information than a graphic document with same amount of carrier data. However, text files are still used because they are easily edited, stored, and transferred [52].

2.8. Experiments Performed *in Silico*

Wang and Zhang [53] have developed a software called WordSpy to detect certain biological features within a genome. This software regards these biological features of a genome as a message hidden in a cover-text of genomic sequences. A Hidden Markov Model is used to decipher the message and to extract over-represented motifs. WordSpy combines word counting and statistical modeling to detect frequently occurring sub-sequences [53].

Since many different coding schemes for inserting messages into DNA have been developed, we decided to develop a software toolkit that would enable us to insert and extract messages from DNA sequences, allow us to compare different coding schemes, and serve as basis for research into developing methods to find and extract messages encoded with unknown coding schemes.

2.9. DNA as Communication Medium

Several papers have been published that discuss the possibility that DNA can be used to communicate with extraterrestrial aliens, or that extraterrestrials have used DNA to communicate with us. Shcerbak and Makukov [54] suggest to use the genetic code itself as an alternative to radio for sending messages to contact extraterrestrial life. Even though the genetic code itself is smaller in capacity than genomic DNA, it is more suitable due to its stronger noise immunity. The flexibility of mapping between codons and amino acids allows modifying the code artificially. The genetic code is the most durable construct known and once fixed, it might stay unchanged over cosmological

timescales. This means that it is exceptionally reliable and therefore a well suited storage medium for an intelligent message, if the biological and thermodynamical conditions are suitable. The answer to the question how life on Earth originated is still not completely certain. The concept of panspermia as hypothesized by Shklovskii and Sagan in 1966 [55] and by Crick and Orgel in 1973 [56] suggests the possibility of life having been seeded on Earth intentionally by extraterrestrial beings. A "signal" in the genetic code with strong statistical characteristics of being artificial in origin would then possibly be the result of such a scenario.

In 1978 Yokoo and Oshima [57] suggested that a civilization more advanced than ours could modify or create a bacterial DNA which could proliferate under favorable circumstances and carry an intelligent message encoded in its base sequence.

Davis [58] developed a method to encode a simple black and white image into a sequence of DNA base pairs, after it had already been converted into binary by Carl Sagan and Frank Drake in 1974 in an attempt to send it as a radio signal from Arecibo, Puerto Rico, to outer space.

2.10. Digital Forensics

Digital Forensics is a relatively new subfield of forensic science. With the beginning of the widespread availability and use of personal computers in the 1980's, the use of computers to perform, hide, or otherwise aid unlawful activity started becoming a serious problem [59]. During the 1980s, digital media was examined directly using non-specialized tools. Forensic tools, hardware as well as software, were first developed in the 1990s. One of the primary goals of digital forensics is to preserve the original data while

collecting evidence. Digital Forensics has the following branches: Computer forensics, database forensics, network forensics, and mobile device forensics.

2.10.1. Computer Forensics

Computer forensics deals with legal evidence found in computers, embedded systems, and digital storage media. Most existing toolkits are proprietary, some are operating system specific. Examples of software toolkits are COFEE, developed by Microsoft, which includes a tool for password decryption, internet history recovery, and other data extraction. It is also able to recover data stored in volatile memory which would be lost if the computer were shut down.

Table 2.13: List of most common computer forensics tools

Name	Platform	License	Description
EnCase [60]	Windows	proprietary	Multi-purpose forensic tool
Sift [61]	Ubuntu	free	Multi-purpose forensic operating system
COFEE [62]	Windows	Only available to law enforcement	A suite of tools for Windows developed by Microsoft
The Sleuth Kit [63]	Unix/Windows	GPL	Large toolkit
Registry Recon[64]	Windows	proprietary	allows users to see how Registries from both current and former installations of Microsoft Windows have changed over time.
The Coroner's Toolkit [65]	Unix	free	Predecessor of The Sleuth Kit

2.10.2. Database Forensics

The least developed branch of digital forensics, Database forensics applies investigative techniques to database content and database related metadata with the goal to identify transactions that indicate evidence of wrongdoing, such as fraud. Investigators examine redo logs, data files and webserver logs to follow the patch of a hacker.

By examining metadata and statistics, investigators could find evidence of database row deletions or the creation of foreign database objects. This may lead to hidden clues that can reveal the path a hacker took. The investigator can then use this information to build a case.

Database forensics tools include LogMiner, which is part of the Oracle database software, and Quisix.

2.10.3. Network Forensics

The goal of network forensics is to capture, record, and analyze network traffic. Tools for Network Forensics include Wireshark, an open source packet analyzer, and SBC, a program that inspects remote access protocols such as SSH, RDP, Telnet, or VNC protocols.

Table 2.14: List of most common network forensics tools

Name	Platform	License	Description
Wireshark [66]	Cross-platform	GPL	Open source packet capture/analyzer
CapAnalysis [67]			performs indexing of data set of PCAP files list of TCP, UDP or ESP streams/flows, passing to the geo-graphical representation of the connections.
OmniPeek [68]	Windows		Packet analyzer
Xplico [69]	Linux	GPL	
Snort[70]	Windows/Linux	GPL	detect probes or attacks, including, but not limited to, operating system fingerprinting attempts, common gateway interface, buffer overflows, server message block probes, and stealth port scans.
NetworkMiner[71]	Windows	proprietary	can be used as a passive network sniffer/packet capturing tool in order to detect operating systems, sessions, hostnames, open ports etc. without putting any traffic on the network. NetworkMiner can also parse PCAP files for off-line analysis and to regenerate/reassemble transmitted files and certificates from PCAP files.

2.10.4. Mobile Device Forensics

Mobile device forensics aims at the recovery of digital evidence from mobile devices with communication ability and internal memory. This includes mobile phones, PDA

devices, GPS devices, and tablet computers. The data targeted by investigators include contact information and photos that are stored on mobile devices. The largest challenge for investigators regarding hardware is the abundance of different connector types.

2.10.5. Anti-Forensics Tools

Just as the developers of digital forensics tools are trying to keep up with the rapidly evolving technology, criminals are developing their own tools as countermeasure to defeat forensics software. [59] It is an "arms race" that is comparable to the cycle of antivirus software versus computer viruses. Two examples of anti-forensics software are Evidence Eliminator, which claims to delete files securely and DECAF, a tool which automatically executes a set of user defined actions on detecting COFEE.

2.11. DNA as a Stegomedium

Criminal organizations are constantly searching for new ways to hide and transmit illegal information, such as child pornography, industrial espionage, and records of other illegal activity. With DNA sequencing and the ability to create artificial DNA sequences becoming increasingly more affordable and more practicable, it could be possible in the near future to store a database containing child pornography or other illegal data inside a DNA database.

When computer forensics examiners investigate evidence in a criminal case, they may not have any reason to modify any evidence files. It is possible to attack hidden content such as stenography and digital watermarks and there are several methods to remove or alter such content with software specifically designed for this purpose.

2.12. Statistics and Artificiality Detection

Benford's Law, which was named after the physicist Frank Benford, refers to the frequency distribution of leading digits in real-life data [72]. It states that 1 occurs as the leading digit approximately 30% of the time, while increasingly larger digits occur in the leading position in decreasing frequency. Benford's law has been found to apply to a wide variety of data sets. It does not only cover the distribution for the first digit, but also for digits beyond that, which approach a uniform distribution. That means it can be generalized from the one leading digit to the n leading digits. There is also a generalization of Benford's law that covers bases other than base 10. It has also been found that Benford's law tends to be most accurate when values are distributed across multiple orders of magnitude.

Benford's law has been used successfully to test an observation that the number of open reading frames and their relationship to genome size differs between eukaryotes and prokaryotes. The main difference is that the former showing a log-linear relationship and the latter a linear relationship [73]. It might be possible to develop an algorithm based on Benford's law to help determine if a DNA sequence contains a hidden message.

2.13. Solving Substitution Ciphers

A substitution cipher is a method of encoding by which units of plaintext are replaced with ciphertext. In simple substitution ciphers, one letter is replaced at a time.

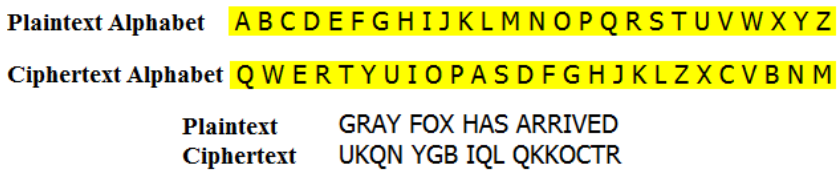


Figure 2.2: Example of a substitution cipher

The methods by which messages are usually encoded in DNA are basically substitution ciphers, e.g. the letter ‘a’ is substituted by the sequence ‘AAA’, the letter ‘b’ by ‘AAC’, and so on. Several different methods have been developed for breaking substitution ciphers. One of our goals is to adapt an algorithm for breaking substitution ciphers to decode a message written in DNA symbols. Almost all approaches use n-grams of letters.

A very promising software called Quipster has been developed by Hasinoff [74]. The software decodes a median of 94% of the cipher letters correctly. The source code is available for download.

A Particle Swarm Optimization (PSO) algorithm has been developed by Uddin and Youssef [29]. Their results show that PSO provides a very powerful tool for the cryptanalysis of simple substitution ciphers using a ciphertext only attack.

Uddin and Youssef [75] also investigated the use of Ant Colony Optimization (ACO) for automated cryptanalysis of classical simple substitution ciphers and found them to be very effective on various sets of encoding keys.

Lucks [76] developed an algorithm which employs an exhaustive search in a dictionary for words that satisfy constraints on word length, letter position and letter multiplicity. His method is not restricted to English and can be used for any language.

It is especially difficult to decode short ciphers, because they have different distribution statistics than larger texts. Hart [77] developed a method that addresses these problems by

using whole words instead of n-grams and by employing a maximum-likelihood estimator.

Jakobsen [56] developed a fast algorithm that is based on a process where an initial key guess is refined through a number of iterations. Each step of this algorithm evaluates the plaintext corresponding to the current key and the result is used as a measure of how close the algorithm is to discovering the correct key. The author claims that only knowledge of the bigram distribution in the ciphertext and the expected bigram distribution in the plaintext is necessary in order to decipher the message. The algorithm currently only uses bigrams, but the author suggests the use of trigrams or whole words for future research.

Forsyth and Safavi-Naini [78] approached the solving of substitution ciphers as a combinatorial optimization problem and developed an algorithm that uses simulated annealing. This algorithm appears very complicated and difficult to implement, but it is very successful at decrypting ciphertexts, especially ones with over 5000 letters.

Peleg and Rosenfeld [79] address this problem as a probabilistic labeling problem and assigned probabilities of representing plaintext letters to every code letter. This was done by using joint letter probabilities. These probabilities were updated in parallel for all code letters, and using this scheme iteratively, they were able to break the cipher.

A genetic algorithm (GA) is a heuristic that is commonly used in artificial intelligence to find useful solutions to search and optimization problems. GAs are a subcategory of evolutionary algorithms which mimic natural evolution using concepts such as inheritance, mutation, selection, and crossover. The genetic algorithm contains a population of strings, referred to as chromosomes, which represent candidate solutions.

Over several generations (iterations of the algorithm), these evolve from a usually randomly generated population to better solutions. The fitness of every individual in the population is evaluated in each generation. Then multiple individuals are selected based on their fitness and recombined and occasionally randomly mutated to form a new population, which is then used in the next iteration of the GA. The GA usually terminates when either a satisfactory fitness level has been reached, or after a maximum number of generations has been created.

Spillman et al. [80] developed a GA for solving substitution ciphers and although they report good results for their algorithm. Delman [81] found GAs to be unreliable for solving substitution ciphers and was unable to reproduce their results.

McClure [82] describes a GA to solve a substitution cipher for 26 characters, which is used in combination with the Wisdom of Artificial Crowds technique. The population size in this approach is 20, with 18 members of the starting population being initialized by creating a random permutation of the English alphabet. The remaining two members were initialized by frequency analysis of the encoded string. In each iteration, the best four members are determined with a fitness score using a dictionary approach and are selected as parents. 75% of the time the first and third members are used as parents of the first child and the second and fourth members become parents of the second child. The remaining 25% of the time, the best population member is copied to create a child.

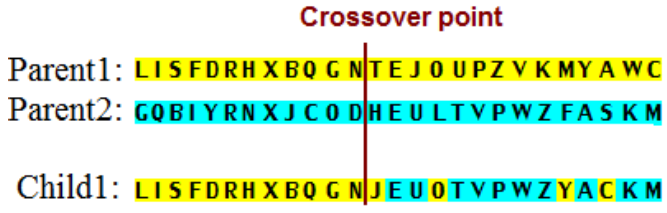


Figure 2.3: Crossover

The crossover is performed by choosing a crossover point between 1 and 26 (the number of characters) at random. All elements before the crossover point were copied from one parent, and all elements after the crossover point, if they did not already exist, were copied from the other parent. Elements that have not been filled in so far are copied from the first parent in the same order as they occur there. The mutation rate is 10% and the number of generations is 10,000.

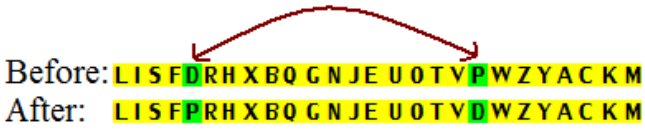


Figure 2.4: Mutation

To determine the fitness of each population member, McClure uses a fitness function that uses the number of incorrectly spelled words. Furthermore, the misspelling is weighted by the number of letters in the word, and also there is a reward for mapping the letters "a", "t", and "e" to the most common characters in the string, since they are the most commonly used letters in the English language. In the equation below, E is the error, S stands for the encoded string, w is the number of words in S, i is a word in S, ni is the number of letters in word i, a is a n by n dimensional binary row vector where the ith location is 0 if word i cannot be found in the dictionary and 1 if it can.

$$E(S) = \sum_{i=1}^{w} a_i\, n_i - [0.12 b_e \sum_{i=1}^{w} n_i] - [0.09 b_t \sum_{i=1}^{w} n_i] - [0.08 b_a \sum_{i=1}^{w} n_i] \quad (2.4)$$

$$b_e = \begin{Bmatrix} 1\ if\ F(k_{20}) \geq 12\% \\ 0\ otherwise \end{Bmatrix} \quad (2.5)$$

$$b_t = \begin{Bmatrix} 1\ if\ F(k_{20}) \geq 9\% \\ 0\ otherwise \end{Bmatrix} \quad (2.6)$$

$$b_a = \begin{Bmatrix} 1\ if\ F(k_{20}) \geq 8\% \\ 0\ otherwise \end{Bmatrix} \quad (2.7)$$

Equation 2 states that b_e is 1 if the letter "e" comprises 12% or more of the decoded text. Equation 3 and 4 are equivalent for the letters "t" and "a" and their respective percentages.

Yampolskiy et al. [83-88] developed Wisdom of Artificial Crowds (WoAC) as a postprocessing algorithm for GA's and Swarm optimization algorithms. It is derived from the Wisdom of Crowds (WoC) algorithm, which is based on the observation that groups are often smarter than the smartest individual in them [89]. For the WoAC algorithm, an $n^x n$ occurrence matrix is constructed. This matrix is used to accumulate the number of times each solution appears. Each row number corresponds to a character while each column number corresponds to the symbol it maps to, in this case letters to letters. The best solution is calculated using the function below:

$$c_{ij} = 1 - I^{-1}_{a_{ij}}(b_1, b_2) \quad (2.8)$$

Where

$$I^{-1}_{a_{ij}}(b_1, b_2) \quad (2.9)$$

is the inverse regularized beta function with parameters b1 and b2 both taking a value of at least 1.

McClure [82] achieved a significant increase in the percentage of correctly identified words in the GA approach over a purely frequency based approach, and a further increase using WoAC.

MCClure [82] performed 8 GA runs for each string and put the best three keys from each run into the voting population for the WoC technique. Four non-repeating

combinations of 12 keys from this population were used to run four WoC tests for each string.

CHAPTER 3

EARLY DESIGN OF SOFTWARE

3.1. Encoding and Inserting Messages

The DNA steganography and steganalysis toolkit we developed has the capability to encode messages in one of several coding schemes and insert them into a DNA sequence (steganography) and for detecting, extracting, and decoding a hidden message from a DNA sequence (steganalysis). Messages can be texts or bitmap images.

This software offers a choice of several different coding schemes. It reads in the coding table for the selected coding scheme from a file and then prompts the user to either type the message to be encoded on the keyboard or to select a file from which the message will be read. Since the ASCII coding scheme is the only one that distinguishes between uppercase and lowercase characters, the program converts all characters in the message into uppercase characters for all coding schemes other than the ASCII coding scheme. The steganography program then encodes the message using the appropriate coding table. The toolkit implements the following coding schemes:

- Huffman code based coding scheme [19]
- Alternating code [19]

- Comma code [12, 19]
- Wong's coding scheme [20]
- Clelland's coding scheme [11]
- DNA-Crypt [10]
- ASCII coding scheme [3]

Coding tables for comma code and alternating code were created arbitrarily, since the original researchers did not provide any.

The message can either be directly written to a file by itself if it is to be stored in a noncoding region, or be inserted into the coding region of an existing DNA sequence file. For inserting a message in a coding region, the algorithm described by Jiao and Gouette [3] is used. DNA sequences can be chosen from a folder where they are stored in FASTA format [24], which is widely used in bioinformatics. The program displays the maximum number of characters a message can have, depending on the coding scheme and the sequence it is to be inserted into.

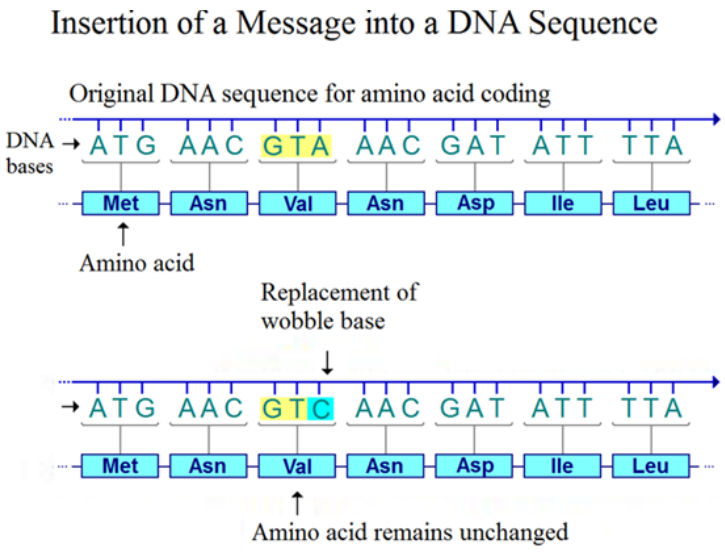


Figure 3.1: Message insertion into coding region

3.2. Approaches to Detecting Messages in DNA

Finding a message that has been inserted into the coding region of a DNA sequence is relatively simple if the original sequence is known. We have developed a program which compares a modified DNA sequence with its original. Since the message is assumed to have been inserted into wobble bases, the first step is to identify wobble base codons in both sequences and to compare them to each other. The first codon where the wobble base is different from the one in the original is identified as the beginning of the message. The last codon where it differs is marked as the end of the message.

The limitation in this approach is that there are codons where the wobble base does not change because it is being replaced by itself. This is not a problem if it happens in the middle of the message. The program therefore assumes it contains one long message instead of several smaller ones. Problems can arise if this happens at the beginning or end of the message, but if the message can be decoded and it is seen that pieces are missing, the program can expanded to go back and fix it.

In order to test the program, the message "THIS IS A TEST" was inserted into the ftsZ DNA sequence using the Wong coding scheme. The program then compared the modified sequence with the original one. It correctly identifies the beginning codon and the end codon of the message and extracts the modified wobble bases.

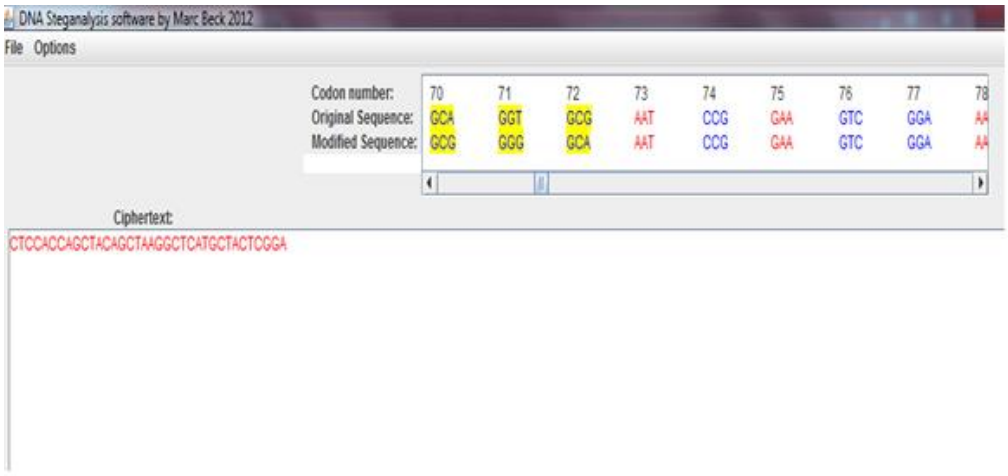


Figure 3.2: Screenshot of message detection software

Finding a message in a noncoding DNA region is much more difficult. But there are ways to determine if a DNA sequence is artificial by statistical analysis. For example, if a certain base is significantly overrepresented, underrepresented, or not present at all, it can be assumed that the sequence is artificial and should be further analyzed to determine if it may contain a message.

Messages that have been encoded using a variation of the alternating code or the comma code are more likely to be detected than messages that were encoded with a different coding scheme. The reason for that is that they have a repeating pattern, which can be detected by a human or a computer program. If every n-th base is the same, this hints at the possibility that comma code or a variation thereof has been used to create this sequence.

One coding scheme that is easy to identify is the DNA-Crypt coding scheme because the low occurrence of As in a message encoded with this scheme.

However, as a countermeasure against attempts to detect messages by counting the occurrence of nucleotides, a coding scheme such as the one developed by Modegi [90] can be used. Modegi's coding scheme uses two codons to encode each letter. Which codon is used is determined by the GC content of the carrier sequence. For example, if a message was to be inserted into a sequence with a high GC content, the letter L would be encoded as CTG, but in a sequence with low GC content it would be TTA [90]. The obvious tradeoff is the number of characters that can be encoded is cut in half.

In order to detect the alternating code, the program stores all odd position characters in one list and all even position characters in another and then compares both of them. If none of the even characters appears in the list with the odd ones and vice versa, the program has detected a message in alternating code with the pattern XYXYXY, where X is either an A or a G and Y a C or a T, or vice versa. The program can easily be extended to detect alternating codes with pattern XXYYXXYY or XXXYYY.

3.3. Extracting Messages from DNA

Since there are four nucleotides, a substitution cipher based coding scheme with a codon length of three, which can encode 64 characters, can be generated in 64! possible ways. And that is only if the same 64 characters are being used. For example, one coding scheme can start with A=AAA, B=AAC, C=AAG,... while another one could be A=AGT, B=CCG, C=CTG,... Brute force guessing which variant has been used to encode the message would take an enormous amount of time and would therefore not be feasible.

We have developed a program for solving simple substitution ciphers where each letter of the English alphabet, numbers from 0-9, and several special characters such as spaces, commas, and periods are each substituted by a combination of three DNA bases. While

normal programs for attacking substitution ciphers search over the space of 26! possible keys, our program has a search space of 64! possible keys. This program is capable of solving all possible coding schemes based on simple substitution ciphers with a codeon length of 3, including the ones developed by Clelland [30] and Wong [3].

The first algorithm we developed uses a list of all the characters in the English alphabet and the frequency of their occurrence in a reference corpus. This corpus contains 801,134 words consisting of 4,899,952 characters, including spaces. The frequency of occurrence in this corpus of the 64 most common characters was recorded. The most common character is the space with 16.2%, followed by the letters E, T, and A with 9.7%, 7.2%, and 6.4%, respectively. It will split the ciphertext into codeons of length 3 and determine the frequency of occurrence of each codeon in the ciphertext. The program will then assign the most frequent letter from our list to the most frequent codon in the ciphertext and so generate a lookup table. Using this lookup table, the program translates the ciphertext into the plaintext. Of course most of the plaintext is still nonsense. After that, the program will split the plaintext into words, using an empty space as a delimiter. It will then compare the words with a dictionary. The dictionary consists of several lists of words, each list contains words with a certain number of letters ranging from two letter words to twelve letter words. If the word matches a word in the dictionary, it will be left alone. If the word differs from a word in the dictionary by a certain number of letters depending on the length of the word, the program will suggest to replace the letters at that particular position by their counterparts in the correct word. For words with a length of four letters or less, the program will only suggest words that differ by one letter. Words

that are five or six letters long will be checked for two letter difference, words with seven or eight letters will be checked for three, and so on.

The following printout is an example how the program suggests replacements for words it doesn't have in its dictionary:

```
GRAF is to be replaced with GRAY by switching F with Y
TLE is to be replaced with THE by switching L with H
TLE is to be replaced with TIE by switching L with I
TLASE is to be replaced with PLACE by switching T with P S with C
TLASE is to be replaced with THOSE by switching L with H A with O
TLASE is to be replaced with CLOSE by switching T with C A with O
TLASE is to be replaced with PLANE by switching T with P S with N
TLASE is to be replaced with TRADE by switching L with R S with D
TLASE is to be replaced with TRACE by switching L with R S with C
TLASE is to be replaced with FLAME by switching T with F S with M
WOS is to be replaced with WAS by switching O with A
WOS is to be replaced with WON by switching S with N
.
.
.
GECEROH is to be replaced with GENERAL by switching C with N O with A H
with L
```

The program will keep track of which letter is suggested to be replaced by which other letter and how many times. It will then switch the codons of the letter pair that has been

suggested for replacement the most often, translate the ciphertext into the plaintext with the updated lookup table and repeat checking the dictionary. With each iteration the number of correct words increases. The program terminates if either no more replacements can be found, or if at least 85% of the words have been correctly identified.

The following printout shows how letters are supposed to be replaced:

```
The system suggested:
replace A with O 13.0 times
replace H with L 5.0 times
replace Y with P 4.0 times
replace L with C 3.0 times
replace I with N 3.0 times
replace L with R 3.0 times
replace C with D 2.0 times
replace S with R 2.0 times
replace S with N 2.0 times
.
.
.
```

Rules:

- A letter cannot be replaced if it occurs at a different position in the same word.
- A letter cannot be replaced if the suggested replacement letter occurs at a different position in the same word.
- A replacement word is to be discarded if another replacement is suggested that requires switching fewer letters.

- a word that has a punctuation character at the end will be checked without the punctuation character.

Another program was written to decode messages that have been encoded using the coding scheme based on the Huffman code [44]. This coding scheme was developed by Smith et al. [38] and uses only the 26 characters of the English Alphabet. There are 16 possible variations of this scheme, based on how the DNA characters are used. Six paragraphs of text of varying length were encoded using the Huffman-based coding scheme for 26 characters as described in Smith et al. Each encoded text was analyzed by a program that counted the frequency of occurrence of single bases, twins, triplets, quadruplets, and quintuplets. The analysis of frequency counts of overall occurrence, as well as quadruplets and quintuplets yielded the most valuable results. It was found that almost always the bases differ noticeably in number of occurrences. Also, in all six paragraphs the least common base never occurs as a quadruplet, and the most common base always occurs as a quintuplet.

Table 3.1: Number of occurrence of certain patterns of bases in ciphertext. Least common in red, most common in green.

Pattern	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
A	299	166	131	126	344	115
C	272	164	107	118	300	110
G	305	183	120	118	343	128
T	293	168	100	122	282	98
AA	104	56	50	42	112	36
CC	82	39	32	27	66	23
GG	75	40	32	26	73	21
TT	32	26	11	18	34	17
AAA	48	18	12	13	26	13
CCC	16	10	6	2	8	6
GGG	37	20	21	19	35	9
TTT	0	1	1	0	0	0
AAAA	17	7	4	1	16	5
CCCC	3	4	0	2	5	3
GGGG	10	6	8	13	19	1
TTTT	0	0	0	0	0	0
AAAAA	2	2	3	1	4	0

This statistical information can be used to identify the coding scheme as the Huffman-based scheme. The bases are then ordered based on frequency of occurrence, with c0 being the most frequent and c3 the least frequent one. The coding table for the Huffman based coding scheme from Smith et al. [38] was hardcoded into the program, replacing each DNA symbol with a variable from c0 through c3. The program then parses the ciphertext and uses the hardcoded table to decode the message.

3.4. Results

The program has been tested using a message the length of a paragraph encoded in Clelland's coding scheme and then with the same message encoded in Wong's coding

scheme. The paragraph has 202 words, 134 not counting repetition. 120 out of those are decoded correctly.

Time : ~10 seconds

With Clelland's coding scheme the program deciphers the text almost correctly, the only errors are that comma and period are switched, and it mistakes the letter J for the number 1. The same errors occur with Wong's coding scheme, but here it also puts a question mark where an apostrophe should be. These errors can easily be fixed by adding more rules.

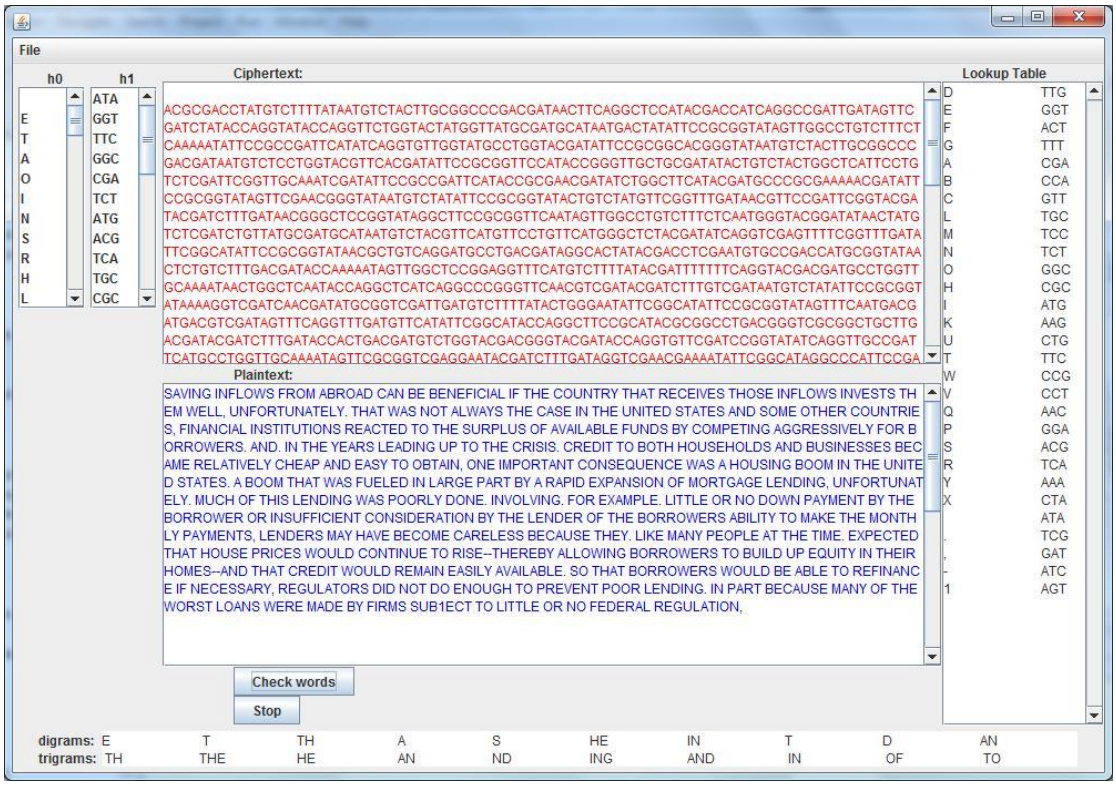


Figure 3.3: Screenshot of decryption software

The program for decoding messages that have been encoded using Huffman-based coding schemes is able to decode messages with all 16 possible variations of the Huffman coding table for 26 characters. The same message as for the previous program was used

for testing this program.

3.5. Limitations

One important question is if the wordlists used in the decoding software should include names and abbreviations (names of people, geographical names, corporations). An early version of the decoding software got into an infinite loop by trying to replace the word 'AND' followed by a punctuation character with the name 'ANDY'. This was resolved by removing 'ANDY' from the wordlist. However, if the name Andy actually occurs in a message, the program might either take longer to decode it correctly, or might not decode it correctly at all. The program also won't know if this name has been correctly decoded. Expanding the wordlist to include every word in the English language could lead to more words being identified as actual words and not garbage strings, but it could also lead to misidentification of words as the above example shows.

How well a message is decoded greatly depends on the length of the message. Messages of a paragraph of 200 words or longer can usually be decoded very quickly and accurately. Messages that are smaller than that, however, usually do not have a character distribution that is close to the character frequency statistic that is being used as a reference.

3.6. Insertion of Media other than Text

Most research in DNA steganography focuses on hiding text and only very little research has been done so far on hiding other media in a DNA sequence. Goldman et al. [12] describe encoding five files of various types in a DNA sequence. These files include a

JPEG 2000 image and a speech in MP3 format. The coding scheme they used utilizes several intermediate steps. First, the image file and the sound file are translated into binary. Then, the text file and the binary data from the other files is translated into a base-3 code and finally into sequences of DNA bases.

Davis [91] describes a method of encoding the black-and-white image of a relatively simple shape (5 by 7 bitmap) into a 35 bit binary sequence, which was then compressed. His approach compares the molecular weights of the bases to obtain an incremental reference. Starting with the smallest base, Cytosine, Davis assigns numbers to the bases in ascending order. This results in C = 1, T =2, A =3, and G =4. This method compresses the binary digits of the bit-mapped image into fewer DNA base symbols by using each base to indicate how many times each binary value (0 or 1) is to be repeated before changing to the respective other value. This technique is widely used in data compression. This can be represented as shown in Table 1. Using this coding method, the thirty-five-bit black-and-white image is translated to only eighteen DNA bases: CCCCCCAACGCGCGCGCT

These can be decoded to yield one of the two following binary sequences:

10101011100010000100001000010000100

or

01010100011101111011110111101111011

This depends on if either a 1 or a 0 is chosen to start the decoding sequence. Transforming either of the two sequences into the correct five-by-seven matrix will produce the image. Since the example used by Davis is bilaterally symmetrical, more

than one of several possible five-by seven matrices will in this case result in producing the correct bitmap [91].

Table 3.2: Coding scheme used by Davis [91]

Base	Bit sequence
C	1 or 0
T	11 or 00
A	111 or 000
G	1111 or 0000

Ailenberg and Rotstein [8] have developed a coding scheme to encode an image that is composed of shapes and their coordinates.

This way of encoding an image is not very efficient. A more feasible approach has been described by Yokoo and Oshima [92]. This approach suggests to arrange the 3-base codons of a DNA sequence in a two dimensional array and then translate one base of each codon into either black or white, with G and C being black and A and T being white, or vice versa. This is done for each base of all the codons, which would result in three separate images.

Hennings and Kettelberger [93] have developed a method to generate music by decoding and transcribing genetic information within a DNA sequence into a music signal having melody and harmony.

We have developed a very similar coding scheme to the one described by Yokoo and Oshima [92], with the difference that we use all three bases of each codon for encoding color information instead of creating three separate images[94]. Our approach will determine the width and height of the array used for creating the image using the two closest factors of the number of codons. This will result in a picture that is as close to a square in shape as possible.

The DNA sequence is arranged in a two dimensional array the same way as described by Yokoo and Oshima [92], but in our case the first base of each codon is used to encode the red portion, the second base for the green portion, and the third for the blue portion of each pixel. DNA bases are translated into RGB values using the following coding table:

Table 3.3: Translation of DNA bases to RGB values

Base	RGB
A	0
C	64
G	128
T	255

Table 3.4: Translation of RGB values into DNA bases

RGB	Base
0-63	A
64-127	C
128-191	G
192-255	T

Each codon encodes one pixel and the coordinates of the codon in the array will be the coordinates of the pixel in the resulting bitmap. The following example shows each step of the encoding process:

DNA sequence:

ATA TAA TAA TAA TTA AAT AAA TTT AAA ATA AAT TTT GAG TTT ATA AAT AAA TTT AAA ATA TAA TTA TTA TTA AAT

DNA sequence as two dimensional array:

ATA	TAA	TAA	TAA	TTA
AAT	AAA	TTT	AAA	ATA
AAT	TTT	GAG	TTT	ATA
AAT	AAA	TTT	AAA	ATA

TAA TTA TTA TTA AAT

The array is created by taking the square root of the number of codons in the DNA sequence. The result is rounded up to give the width and rounded down to give the height of the image. The two numbers are multiplied and if the result is less than the number of codons, the smaller number is increased by 1. This will result in an array that is large enough for all codons, in some cases slightly larger. The extra space will be filled with white pixels in the resulting image.

After translation into RGB:

0,255,0	255,0,0	255,0,0	255,0,0	255,255,0
0,0,255	0,0,0	255,255,255	0,0,0	0,255,0
0,0,255	255,255,255	127,0,127	255,255,255	0,255,0
0,0,255	0,0,0	255,255,255	0,0,0	0,255,0
255,0,0	255,255,0	255,255,0	255,255,0	0,0,255

Figure 3.4: Resulting image (enlarged by factor 16)

This method allows the encoding of 64 colors and ensures that the encoding of all the most common colors such as red, green, blue, yellow, magenta, orange, grey, black, and white is possible. The use of only 64 colors obviously leads to the loss of color information. Also, with the current algorithm the program assumes that the width and height of an image are as similar (a square, or approximately a square) as possible. For example, a 120x40 pixel image would be decoded as a 60x80 pixel image. A possible solution would be to encode the dimensions of the image as well. Our method is simpler and more storage space efficient than the one described by Goldberg [6], but as a tradeoff

can encode fewer colors. It is also more specialized toward images, while Goldberg's approach is geared toward a variety of data types. Further research could lead to the development of algorithms to detect, extract and decode images that have been hidden in DNA sequences [94]. These methods could be used for forensic purposes. Similar algorithms have already been developed for text-based DNA Steganalysis [25].

CHAPTER 4

FINAL DESIGN OF SOFTWARE

4.1. Overview

The main menu allows the user to choose between extraction of text, insertion of text, or insertion/extraction of images. If the user chooses to extract a message, the message can be loaded from a txt file or a FASTA file. Header information from FASTA files is automatically removed. The user has the choice between two encoding algorithms: the Dictionary approach and the GA/WoAC approach. Individual settings for each algorithm can be defined by the user. These include population size, number of iterations, and stop condition in case of the GA. It is also possible to decode a message without extracting it from a DNA sequence. This message can either be loaded from a file, or typed or pasted into the appropriate text field. The basic coding scheme is determined before attempting to decode any message in order to choose the codeon size, and in case of a Huffman code base scheme, start the appropriate decoding algorithm.

For message insertion, the user can load a cover sequence from file, and again the header is stripped if the file is in FASTA format. The message to be inserted can be typed or loaded from a file. The user can choose between several coding schemes to encode the

message, and the program will alert the user if there are not enough wobble bases within coding regions to accommodate a message of this particular length.

A lookup table on the side of the screen shows how the current coding scheme translates between DNA codeons and alphanumeric characters.

The image insertion and extraction feature is fairly straightforward and does not require any detailed explanation.

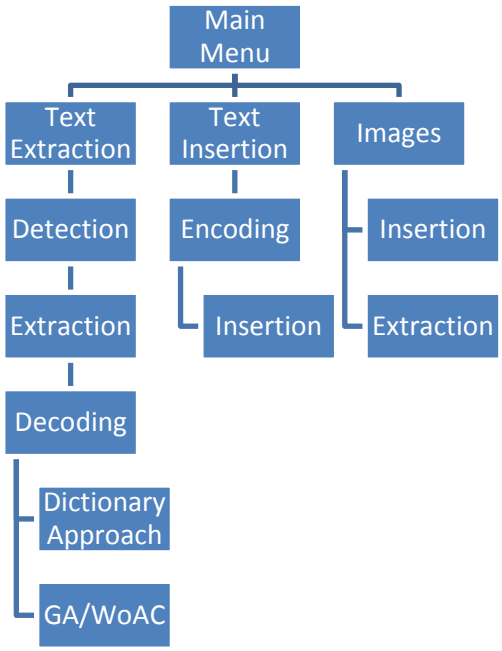


Figure 4.1: Organizational chart of DNA Steganalysis Software

4.2. Genetic Algorithm and Wisdom of Artificial Crowds

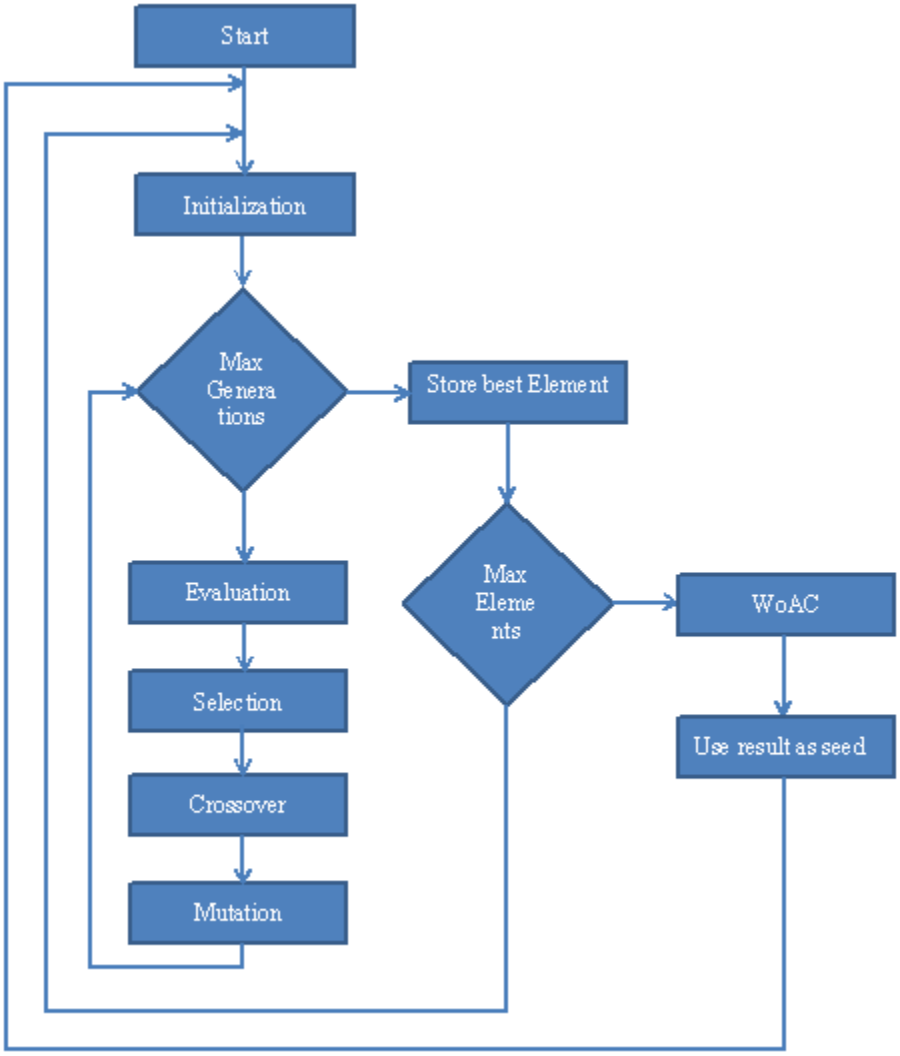


Figure 4.2: Flow chart of GA with WoAC

In order to increase the accuracy with which shorter messages can be decoded we began to search for alternative methods to the dictionary approach. One of the proposed alternatives is to use a Genetic Algorithm (GA). The GA developed by McClure [95] was modified to accommodate a larger alphabet [96].

Both the dictionary approach and the GA with WoAC approach were tested with two different sample messages of different lengths. The first has 202 words, 134 not counting

repetition. The second message has 51 words, of which 35 are unique. Both messages have been encoded with the coding scheme developed by Clelland [1], with the Comma code, and with the Alternating code.

Our software package was written in Java 7 using Eclipse v.4.4.0. The computer used for this experiment has an Intel Core i7 processor and 10 GB RAM and runs Windows 7 Home Premium 64 bit.

The settings for the GA are as follows: 20 population members, 5000 generations, and a mutation rate of 10%. The results of 10 runs of the GA were entered into the WoAC. Then the results of the WoAC were used to initialize the GA for the next 10 runs, with their results entering into the WoAC again. For the shorter text the GA was run with 1000 generations.

The keys we produced with ten runs of the GA were fed into the WoAC algorithm. Then we use the key obtained from WoAC as seed value to initialize two out of twenty population members in another run of the GA. The remaining 18 population members are initialized at random.

In order to be able to work with 64 characters instead of 26 we counted the frequency of occurrence of the 64 most common symbols and characters in our sample corpus and adjusted the formula accordingly. Since we take spaces and punctuation into account, our most common character is now the space with 16%, followed by the letters e, t, and a with 9%, 7%, and 6%, respectively. Also, besides rewarding high percentage of occurrence of the most frequent characters, we punish high percentage of occurrence of the 14 least frequent characters. The dictionary used in both approaches contains over 28,000 words.

Table 4.1: Changes compared to McClure

	McClure	**Our Algorithm**	
Character	Reward if	Reward if	Punish if
[space]	----------	>=16%	----------
e	>=12%	>=9%	----------
t	>=9%	>=7%	----------
a	>=8%	>=6%	----------
+	----------	----------	<1%
*	----------	----------	<1%
=	----------	----------	<1%
_	----------	----------	<1%
$	----------	----------	<1%
&	----------	----------	<1%
#	----------	----------	<1%
<	----------	----------	<1%
>	----------	----------	<1%
^	----------	----------	<1%
%	----------	----------	<1%
@	----------	----------	<1%
[	----------	----------	<1%
]	----------	----------	<1%

The GA is able to decode the first sample message with 100% accuracy independent of the coding scheme; however each GA run takes an average of 32 minutes and the total decoding process takes almost 6 hours. The shorter message is decoded in about 7 minutes with 89% accuracy.

Table 4.2: Results of decoding messages with the dictionary approach.

Decoded with dictionary approach				
Coding scheme	**Words total**	**Words correct**	**Characters correct**	**Time**
Clelland	202	89%	76%	0.7 sec
Clelland	51	51%	52%	0.06 sec
Comma	202	99%	80%	0.8 sec
Comma	51	64%	42%	0.08sec
Alternating	202	99%	80%	0.9 sec
Alternating	51	58%	50%	0.07 sec

Table 4.3: Results of decoding messages with the GA/WoAC approach.

Decoded with Genetic Algorithm/ Wisdom of Artificial Crowds				
Coding scheme	**Words total**	**Words correct**	**Characters correct**	**Time**
Clelland	202	100%	93%	5hrs 40 min
Clelland	51	89%	74%	7 min 6 sec
Comma	202	100%	96%	5 hrs 33 min
Comma	51	74%	50%	6 min 57 sec
Alternating	202	100%	96%	5hrs 48 min
Alternating	51	84%	53%	7 min 13 sec

The table below shows the accuracy for each GA run compared to each other and to the WoAC for the long message that has been encoded with Clelland's coding scheme. The results for decoding the messages with the other coding schemes are similar. The table also contains the results of the GA runs 11-20 which use the result from the WoAC as key. The end result is obtained by using the WoAC algorithm on runs 11-20. Some of the GA iterations actually produce worse results individually than the dictionary approach. Words correct means words that can be actually found in the dictionary. The first 10 iterations are independent from each other, as are the last 10.

Table 4.4: Comparing the seperate GA runs to the WoAC results.

Iteration	Words correct	Characters Correct
1	99%	90%
2	98%	86%
3	84%	79%
4	10%	34%
5	10%	34%
6	43%	59%
7	73%	69%
8	9%	28%
9	9%	0%
10	96%	79%
WoAC	99%	86%
11	99%	83%
12	99%	83%
13	98%	79%
14	100%	93%
15	100%	93%
16	100%	93%
17	100%	93%
18	99%	90%
19	100%	93%
20	100%	93%
End result	100%	93%

As we can see in run 9 we can even get words that are in the dictionary when all the characters are switched. Because of so many “correct” words, the key generated has such a good error score that it gets stuck in a local maximum [96].

4.3. Determining Coding Schemes

At the beginning the steganalysis tool could only decode messages that have been encoded with substitution cipher coding schemes such as the ones developed by Clelland et al.[30], Wong et al.[3], and variations thereof, as well as messages that have been encoded with variations of the Huffman-based coding scheme for 26 characters

developed by Smith et al.[38]

The next step was to add the capability to decode messages encoded in other coding schemes that have been described previously, such as the alternating code or comma code. In order to do so, we added a feature to determine which coding scheme was most likely used to decode the message. This was possible by making use of unique properties of those coding schemes.

An unusual property of the alternating code that can be used to determine if a message has been encoded with the alternating code is the following: in a given piece of message DNA, the number of G:C pairs will be the same as the number of A:T pairs [38].

Another, possibly more accurate method to determine if a message has been encoded in the alternating code is to first turn all A's and G's in the ciphertext into R's, and all C's and T's into Y's. Then to split the ciphertext into codons of length 6 and check if the first codon matches any of the following patterns: RYRYRY,YRYRYR,YYRRYY,RRYYRR,YYYRRR,RRRYYY. If it does, the program would check if the remaining codons follow the same pattern.

Once the Alternating code has been detected, it will be decoded the same way as simple substitution schemes, but with a codon length of 6. The pattern only plays a role in detection, but not in decryption.

The DNA symbol which is the comma character fulfills all of the following three criteria: It is the first and the last character of the sequence, it always appears by itself, never as part of a twin, triplet, etc. and the distance between instances of the comma symbol is always the same.

In order to determine if a message has been encoded in comma code, our program checks which one of the four DNA bases fulfills all three of these criteria. If a base is found that meets those criteria is found, this base is then assigned as the comma character, and the distance between each occurrence of this comma character +1 is the codon length.

This software should be able to detect any variation of the comma code, no matter which DNA base is the comma character, and no matter the codon length. It therefore can detect both the version described by Smith et al. [38], and the one by Brenner et al. [29], as well as other variations of the comma code. Before decoding the comma code, the comma character at the end of the ciphertext is removed, since it adds no useful information to the message.

When a message encoded with the coding scheme developed by Arita et al. [29] is inserted into a coding region of a DNA sequence, a 0 indicates to leave the 3rd base of a codon unchanged, while a 1 indicates that it needs to be changed. This means that the ratio of changed wobble bases/unchanged wobble bases within message should be around ½, while with other coding schemes it should be around ¼. This is used to detect a message encoded with this particular coding scheme

The decoding of the Arita coding scheme will be done in two steps: DNA to binary and then binary to text. Since there are only 16 possibilities, the decoding from DNA to binary is done in a brute force approach. After each decoding attempt, the resulting binary sequence is broken into strings of length 5 and each string is checked for the parity bit. If the number of ones is even, the parity bit is one, if it is odd the parity bit is zero. If the parity bit computes correctly for more than 95% of the message, it is decoded from

binary to plaintext, otherwise the next decoding attempt is tried. The decoding from binary to plaintext is accomplished in a similar manner as with the substitution schemes.

The ASCII and Yachie coding schemes are so similar that they can be decoded with the same algorithms.

The algorithms that have been developed for the basic Huffman based coding scheme can be modified to work for the improved Huffman coding scheme. The only difference other than the number of characters and with which symbols they are encoded is the division of characters into three subgroups and the use of a header codon to select the subgroup.

4.4. Message Extraction

Since the length of the message is not known, the extraction algorithm currently extracts all wobble bases within the coding regions of the cover sequence. However, the message may only use a small part of the available space, which will result in a great amount of garbage information at the end of the actual message. If the message uses only a small percentage of the available coding space, the amount of garbage information will be very large. This will prevent both decoding algorithms from working properly, since they both depend on the frequency of occurrence of characters.

This problem is addressed by attempting to decode only a part of the message, a percentage that can be selected by the user. Once the coding scheme is known, the entire message can then be decoded using that coding scheme and the garbage information at the end can be discarded. The only problem with that approach is that it only works for messages that have been inserted at the beginning of the DNA sequence.

4.5. Integration of Components

All parts of the DNA steganography and steganalysis toolkit have been integrated using a common graphical user interface (GUI) and a common file structure. Also, the software creates a report which is automatically saved in a file once a message has been detected and decoded. This report contains detailed information at which position in the DNA sequence the message begins, where it ends, its length, which coding scheme was used, a coding table, as well as frequency counts of single bases, bigrams, trigrams, and quadrigrams of bases. The goal of this effort is to keep our software toolkit user friendly, flexible, and expandable.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1. Conclusion

DNA steganography and DNA steganalysis are fairly new research fields and therefore they offer many opportunities to improve upon existing approaches for steganography as well as steganalysis. This project combines several approaches for DNA steganography undertaken by several independent groups of researchers in the past, and builds upon the results of their work to create software for steganalysis of the coding schemes they developed. There has been a great amount of research done in the resent past on hiding messages in DNA, but not much on finding hidden messages. This project aims at covering a variety of different approaches for DNA steganography. The GA clearly takes several orders of magnitudes more time, but is able to decode short messages at greater accuracy than the dictionary approach. Both methods complement each other, while the dictionary approach is faster; the GA is more accurate and also performs better at decoding shorter messages. Shorter messages need more generations but take less time. The WoAC algorithm used in combination with multiple runs of the GA provides clearly improved results compared to the individual runs of the GA by themselves. The end result is even better after using the WoAC results as seed for the next 10 GA runs. Both

algorithms are currently being improved further and tested with a greater variety of messages, such as messages with a large amount of numbers and special characters and messages that contain names and foreign words. The key is to make this software toolkit as flexible as possible, so it can be adapted in the future to deal with new coding schemes and new approaches to hide messages.

5.2. Future Work

5.2.1. Improvements

Both message decoding approaches still have room for improvement and it is possible to further increase their accuracy. The GA could possibly be improved to converge faster by experimenting with different crossover and mutation algorithms. An idea would be to increase the mutation rate if too many population members are similar, and then decrease it over generations.

It would be interesting to see how both algorithms perform when attempting to decode messages that have been encrypted before being encoded.

We have written a program that creates a dictionary and calculates the frequency of occurrence of characters from a sample corpus. This program can be easily modified to create a dictionary and a frequency count based on a specific topic. The program for encoding and inserting messages will have an option to encrypt a message with a shift cipher before encoding it. There will also be a function to decrypt messages that have been encoded in such manner. The encoding software can be improved to allow the user to determine at which location a message should be inserted into a DNA sequence, given a sequence long enough to allow that. Currently messages are inserted beginning with the

first wobble base codon of the sequence. In turn, the message detection program will also be modified to be able to detect a message at any given location inside a DNA sequence. The program will then also be able to analyze a DNA sequence using different reading frames as well as backwards, and inverse. We will also expand our system by enabling it to detect messages that have been encoded with different coding schemes in the same DNA sequence. There is also the possibility to include an option to either use or ignore Start- and Stop codons.

Furthermore, the program for decoding Huffman based coding schemes can be expanded to be able to decode messages with coding schemes for more than 26 characters. It should therefore be upgraded to be able to handle the improved Huffman code based scheme developed by Ailenberg and Rotstein [25].

Both of the decoding methods did not perform well in decoding numbers and punctuation characters. One solution is to create rules that will assign higher fitness scores in the GA to certain patterns such as dates. In order to decode punctuation characters more accurately, there is the possibility to run a post processing function after the letters and numbers have been decoded to ensure certain rules, for example that a period or comma is usually followed by an empty space, or that there is a closing parenthesis for every opening parenthesis. Another possibility to address this problem is to develop an algorithm based on Benford's law.

The WordSpy algorithm developed by Wang and Zhang [28] could also be investigated as a way to detect new coding schemes that do not already have specialized algorithms for detecting and decoding.

Once the software has detected a message, it would automatically decide based on certain characteristics which approach to use in attempting to decode a message. A decision tree needs to be developed to determine what the program should do next in case an approach to decode the message fails.

There are currently two decoding algorithms, the dictionary approach and the GA. Thanks to its modular nature the software toolkit can be improved by adding other decoding algorithms. It would then be possible to attempt to decode a message with each algorithm and feed the resulting keys from each algorithm into the WoAC.

A way to speed up the GA/WoAC approach could be to modify the program to run multiple instances of the GA in parallel since they are all independent from each other. This could be accomplished at first on a multi core computer, and if successful, later on a larger scale on a cluster. Of course, the speedup cannot be expected to be linear, because parts of the program, such as the initialization and the WoAC cannot be parallelized. A more accurate method to measure the time each of the two approaches takes needs to be implemented. This can be accomplished by counting clock cyles instead of CPU time.

The observation that samples of DNA sequences of the same genome are significantly more similar to each other than to those of sequences from other organisms [97] can help detect the presence of a message. Also, certain bi-,tri,- and quadrigrams of DNA bases occur more often in certain genomes than in others. For example, Burge et al. [98] have discovered that the bigram CG is strongly underrepresented in vertebrates and mitochondrial genomes. The problem here is that a hidden message has to be very large in order to significantly affect the distribution of base n-grams. Message detection can be improved by using statistical analysis. In order to obtain a statistical baseline, 10

variations of the human genome could be downloaded from an online database and the frequency of occurrence of n-grams would then be counted, ranging from n=1 to n=6. Messages will be inserted and the statistics of the modified sequences will be compared to the statistics of the original sequences. Depending on the results, an algorithm could then be derived to determine with a certain percentage of accuracy if a DNA sequence contains a message.

Other Applications

With rapid advances in genetic engineering and in DNA sequencing, many new research applications for our software become available in the near future. Elements of this software toolkit can be modified and used for other purposes in bioinformatics as well as in digital forensics. It is also possible for the software to be used to detect natural occurring patterns instead of artificial messages.

A research team at the University of Washington discovered that a certain group of codons, which are called duons, can have two functions, the first controls protein sequencing, and the second is partially responsible for gene control. Both functions appear to have evolved in concert with each other. The gene control instructions seem to aid in the stabilization of certain beneficial features of proteins and how they are made. This fact means that many changes in the DNA that seem to change protein sequences may actually cause disease by disrupting functions responsible for gene control or possibly even both mechanisms at the same time [99]. This could possibly also mean that inserting messages in the coding regions of DNA sequences may have an effect on the carrier organism after all. Different methods for inserting messages into DNA sequences

would need to be developed in order to circumvent this problem. Our software is modular and can easily be adapted to address these challenges.

REFERENCES

[1] C. T. Clelland, V. Risca, and C. Bancroft, "Hiding messages in DNA microdots," *Nature,* vol. 399, pp. 533-534, 1999.

[2] B. Shimanovsky, J. Feng, and M. Potkonjak, "Hiding Data in DNA," *Lecture Notes in Computer Science,* vol. 2578, pp. 373-386, 2003.

[3] P. C. Wong, K.-K. Wong, and H. Foote, "ORGANIC DATA MEMORY Using the DNA Approach," *Communications of the ACM,* vol. 46, pp. 95-98, 2003.

[4] K. Tanaka, A. Okamoto, and I. Saito, "Public-key system using DNA as a one-way function for key distribution," *Biosystems,* vol. 81, pp. 25-9, Jul 2005.

[5] N. Yachie, K. Sekiyama, J. Sugahara, Y. Ohashi, and M. Tomita, "Alignment-Based Approach for Durable Data Storage into Living Organisms," *Biotechnology Progress,* vol. 23, p. 4, 2007.

[6] D. Heider and A. Barnekow, "DNA-based watermarks using the DNA-Crypt algorithm," *BMC Bioinformatics,* vol. 8, p. 176, 2007.

[7] S.-H. Jiao and R. Goutte, "Hiding data in DNA of living organisms," *Natural Science,* vol. 01, pp. 181-184, 2009.

[8] M. Ailenberg and O. Rotstein, "An improved Huffman coding method for archiving text, images, and music characters in DNA," *Biotechniques,* vol. 47, pp. 747-54, Sep 2009.

[9] D. G. Gibson, J. I. Glass, C. Lartigue, V. N. Noskov, R. Y. Chuang, M. A. Algire*, et al.*, "Creation of a bacterial cell controlled by a chemically synthesized genome," *Science,* vol. 329, pp. 52-6, Jul 2 2010.

[10] H. Mousa, K. Moustafa, W. Abdel-Wahed, and M. Hadhoud, "Data Hiding Based on Contrast Mapping," *The International Arab Journal of Information Technology,* vol. 8, pp. 147-154, 2011.

[11] G. M. Church, Y. Gao, and S. Kosuri, "Next-Generation Digital Information Storage in DNA," *Sciencexpress,* vol. 337, p. 1628 2012.

[12] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos*, et al.*, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature,* vol. 494, pp. 77-80, Feb 7 2013.

[13] "Dr. Ido Bachelet Talk on Bionic Technologies," in *Personality Cafe* vol. 2015, ed, 2015.

[14] J. D. Watson and F. H. C. Crick, "Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid," *Nature,* vol. 171, pp. 737-738, 1953.

[15] B. Anam, K. Sakib, A. Hossain, and K. Dahal, "Review on the Advancements of DNA Cryptography," presented at the International conference on Software, Knowledge, Information Management and Application, Paro, Bhutan, 2010.

[16] R. G. Martin, J. H. Matthaei, O. W. Jones, and M. W. Nirenberg, "Ribonucleotide composition of the genetic code," *Biochemical and biophysical research communications,* vol. 6, pp. 410-414, 1962.

[17] S.-H. Jiao and R. Goutte, "Code For Encryption Hiding Data Into Genomic DNA," in *International Conference on Software Process*, 2008.

[18] E. C. Hayden, "The $1000 genome," *Nature,* vol. 507, pp. 294-295, 2014.

[19] K. Wetterstrand. (2014, 03/03). *Data from the NHGRI Genome Sequencing Program (GSP)*.

[20] B. Mole. (2014, 03/07). *The Gene Sequencing Future is here*.

[21] L. M. Adleman, "Molecular Computation Of Solutions To Combinatorial Problems," *Science, New Series,* vol. 266, pp. 1021-1024, 11/11/1994 1994.

[22] M. Ogihara and A. Ray, "Simulating Boolean Circuits on a DNA Computer," in *RECOMB*, 1997, pp. 226-231.

[23] Y. Benenson, B. Gil, U. Ben-Dor, R. Adar, and E. Shapiro, "An Autonomous Molecular Computer for logical control of Gene Expression," *Nature,* vol. 429, pp. 423-429, 2004.

[24] C. M. Bogard, E. C. Rouchka, and B. Arazi, "DNA media storage," *Progress in Natural Science,* vol. 18, pp. 603-609, 2007.

[25] Y. Amir, E. Ben-Ishay, D. Levner, S. Ittah, A. Abu-Horowitz, and I. Bachelet, "Universal computing by DNA origami robots in a living animal," *Nat Nanotechnol,* vol. 9, pp. 353-7, May 2014.

[26] H. Weinzierl, "Digital Universe Invaded By Sensors," ed. HOPKINTON, MASS: EMC, 2014.

[27] S. Shrivastava and R. Badlani, "Data Storage in DNA," *International Journal of Electrical Energy,* vol. 2, pp. 119-124, 2014.

[28] M. Arita and Y. Ohashi, "Secret Signatures Inside Genomic DNA," *Biotechnology Progress,* vol. 20, pp. 1605-1607, 2004.

[29] M. Arita, "Comma-free design for DNA words," *Communications of the ACM,* vol. 47, p. 99, 2004.

[30] C. T. Clelland, V. Risca, and C. Bancroft. (1999) Hiding messages in DNA microdots. *Nature*. 533-534.

[31] S. Brenner, S. R. Williams, E. H. Vermaas, T. Storck, K. Moon, C. McCollum*, et al.*, "In vitro cloning of complex mixtures of DNA on microbeads: Physical separation of differentially expressed cDNAs," *Proceedings of the National Academy of Sciences of the United States of America,* vol. 97, pp. 1665-1670, 02/15/2000 2000.

[32] D. Heider and A. Barnekow, "DNA watermarks: a proof of concept," *BMC Mol Biol,* vol. 9, p. 40, 2008.

[33] C. Bancroft, T. Bowler, B. Bloom, and C. T. Clelland, "Long-Term Storage of Information in DNA," *Science, New Series,* vol. 293, pp. 1763-1765, 09/07/2001 2001.

[34] Anonymous, "A Y3K bug," *nature biotechnology,* vol. 18, 2000.

[35] T. Siebert, "CDs Are Not Forever: The Truth About CD/DVD Longevity, “Mold” & “Rot”," in *makeuseof* vol. 2015, ed, 07/03/2012.

[36] J. Aron, "DNA in glass-the ultimate archive," *New Scientist,* vol. 225, p. 15, 02/14/2015 2015.

[37] J. W. Drake, B. Charlesworth, D. Charlesworth, and J. F. Crow, "Rates of Spontaneous mutation," *Genetics,* vol. 148, p. 20, 1998.

[38] G. C. Smith, C. C. Fiddes, J. P. Hawkins, and J. P. L. Cox, "Some possible codes for encrypting data in DNA," *Biotechnology Letters,* vol. 25, pp. 1125-1130, 2003.

[39] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal,* vol. 29, pp. 147-160, 1950.

[40] A. S. Tanenbaum, *Computer Networks*, 4th edition ed. New York, NY, USA: Prentice Hall, 2002.

[41] R. Soni, V. Soni, and S. K. Mathariya, "Innovative field of cryptography: DNA cryptography," in *First International Conference on Information Technology Convergence and Services (ITCS 2012)*, Bangalore, India, 2012, pp. 161-179.

[42] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for Data Hiding," *IBM Systems Journal,* vol. 35, pp. 313-336, 1996.

[43] M. Nirenberg, "Historical review: Deciphering the genetic code – a personal account," *Trends in Biochemical Sciences,* vol. 29, pp. 46-54, 2004.

[44] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE,* pp. 1098–1102, 1952.

[45] S. Singh, *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots to Quantum Cryptography*. New York: Doubleday, 1999.

[46] C. T. Clelland, V. Risca, and C. Bancroft, "Hiding messages in DNA microdots.pdf," *Nature,* vol. 399, pp. 533-534, 1999.

[47] S.-H. Jiao, "Hiding data in DNA of living organisms," *Natural Science,* vol. 01, pp. 181-184, 2009.

[48] Q. Tang, G. Ma, W. Zhang, and N. Yu, "Reversible Data Hiding for DNA Sequences and Its Applications," *Digital Crime and Forensics,* vol. 6, 2014.

[49] D. C. Jupiter, T. A. Ficht, Q.-M. Qin, and P. de Figueiredo, "DNA Watermarking of Infectious Agents Progress and Prospects," *Public Library of Science Pathogens,* vol. 6, pp. 1-3, 2010.

[50] N. F. Johnson and S. Jajodia, "Steganalysis: The Investigation of Hidden Information," in *IEEE Information Technology Conference*, Syracuse, NY, 1998, pp. 113 - 116

[51] B. Li, J. Huang, and Y. Q. Shi, "Steganalysis of YASS," *IEE Transactions on Information Forensics and Security,* vol. 4, pp. 369 - 382 Sept. 2009 2009.

[52] S. Xin-guang, L. Hui, and Z. Zhong-liang, "A Steganalysis Method Based on the Distribution of Characters," in *8th International Conference on Signal Processing*, Beijing, China, 2006.

[53] G. Wang and W. Zhang, "A steganalysis-based approach to comprehensive identification and characterization of functional regulatory elements," *Genome Biol,* vol. 7, p. R49, 2006.

[54] V. I. shCherbak and M. A. Makukov, "The “Wow! signal” of the terrestrial genetic code," *Icarus,* vol. 224, pp. 228-242, 2013.

[55] I. Shklovskii and C. Sagan, *Intelligent life in the universe*. New York: Dell, 1966.

[56] F. H. C. Crick and L. E. Orgel, "Directed Panspermia," *Icarus,* vol. 19, pp. 341-346, 1973.

[57] H. Yokoo and T. Oshima, "Is Bacteriophage X174 DNA a Message from Extraterrestrial Intelligence?," *Icarus,* vol. 38, 1978.

[58] J. Davis, "Microvenus," *Art Journal,* vol. 55, pp. 70-74, 1996.

[59] M. Reith, C. Carr, and G. Gunsch, "An Examination of Digital Forensic Models," *International Journal of Digital Evidence,* vol. 1, 2002.

[60] EnCase. (04/01/2015). *EnCase eDiscovery V5*. Available: https://www.guidancesoftware.com/products/Pages/encase-ediscovery/overview.aspx

[61] D. F. a. I. response. (04/01/2015). *SANS Investigative Forensic Toolkit (SIFT) Workstation Version 3*. Available: http://digital-forensics.sans.org/community/downloads/

[62] Microsoft. (04/01/2015). *Microsoft Cofee*. Available: https://cofee.nw3c.org/

[63] (04/01/2015). *The Sleuth Kit*. Available: http://www.sleuthkit.org/sleuthkit/

[64] A. Recon. (04/01/2015). *RegistryRecon*. Available: http://www.arsenalrecon.com/apps/recon/

[65] "The Coroner's Toolkit (TCT)."

[66] Wireshark. (2015, 04/01/2015). *About Wireshark*. Available: https://www.wireshark.org/about.html

[67] CapAnalysis. (04/01/2015). *CapAnalysis*. Available: http://www.capanalysis.net/ca/#about

[68] WildPackets. (2015, 04/01/2015). *OmniPeek Network Analyzer*. Available: http://www.wildpackets.com/products/omnipeek_network_analyzer

[69] Xplico. (04/01/2015). *Open Source Network Forensic Analysis Tool (NFAT)* Available: http://www.xplico.org/

[70] Snort. (04/01/2015). *Snort*. Available: https://www.snort.org/

[71] Netresec. (04/01/2015). *NetworkMiner*. Available: http://www.netresec.com/?page=NetworkMiner

[72] F. Benford, "The Law of Anomalous Numbers," *Proceedings of the American Philosophical Society,* vol. 78, pp. 551-572, 1938.

[73] J. L. Friar, T. Goldman, and J. Perez–Mercader, "Genome Sizes and the Benford Distribution," *PLoS One* vol. 7, 2012.

[74] S. Hasinoff, "Solving Substitution Ciphers," *A Technical Report, University of Toronto,* 2003.

[75] M. F. Uddin and A. M. Youssef, "An Artificial Life Technique for the Cryptanalysis of Simple Substitution Ciphers," presented at the CCECE+CCGEI, Ottawa, Canada, 2006.

[76] M. Lucks, "A Constraint Satisfaction Algorithm for the Automated Decryption of Simple Substitution Ciphers," in *CRYPTO '88*, Santa Barbara, California, USA, 1988, pp. 132–144.

[77] G. W. Hart, "To decode short Cryptograms," *Communications of the ACM,* vol. 37, pp. 102-108, 1994.

[78] W. S. Forsyth and R. Safavi-Nani, "Automated Cryptanalysis of substitution ciphers," *Cryptologia,* vol. 17, pp. 407-424, 1993.

[79] S. Peleg and A. Rosenfeld, "Breaking Substitution Ciphers Using a Relaxation Algorithm," *Communications of the ACM,* vol. 22, pp. 598–605, 1979.

[80] R. Spillman, M. Janssen, B. Nelson, and M. Kepner, "Use of a Genetic Algorithm in the Cryptanalysis of Simple Substitution Ciphers," *Cryptologia,* vol. 17, pp. 31-44, 1993.

[81] B. Delman, "Genetic Algorithms in Cryptography," Master of Science in Computer Engineering, Department of Computer Engineering, Rochester Institute of Technology, Rochester, New York, 2004.

[82] P. McClure, "Title," unpublished|.

[83] R. V. Yampolskiy and A. El-Barkouky, "Wisdom of artificial crowds algorithm for solving NP-hard problems," *International Journal of Bio-Inspired Computation,* vol. 3, pp. 358-369, 2011.

[84] A. Ben Kalifa and R. V. Yampolskiy, "GA with Wisdom of Artificial Crowds for Solving Mastermind Satisfiability Problem," presented at the Int. J. Intell. Games & Simulation, 2011.

[85] L. H. Ashby and R. V. Yampolskiy, "Genetic algorithm and Wisdom of Artificial Crowds algorithm applied to Light up," presented at the Computer Games (CGAMES), 2011 16th International Conference on, Louisville,KY, 2011.

[86] R. V. Yampolskiy, L. H. Ashby, and L. Hassan, "Wisdom of Artificial Crowds-A Metaheuristic Algorithm for Optimization," *Journal of Intelligent Learning Systems & Applications,* vol. 4, 2012.

[87] A. C. Port and R. V. Yampolskiy, "Using a GA and Wisdom of Artificial Crowds to solve solitaire battleship puzzles," presented at the Computer Games (CGAMES), 2012 17th International Conference on, 2012.

[88] R. Hughes and R. V. Yampolskiy, "Solving Sudoku Puzzles with Wisdom of Artificial Crowds," *International Journal of Intelligent Games & Simulation,* vol. 7, 2013.

[89] J. Surowiecki, *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*. New York Doubleday, 2004.

[90] T. Modegi, "Watermark Embedding Techniques for DNA Sequences Using Codon Usage Bias Features," presented at the 16th International Conference on Genome Informatics, Yokohama, Japan, 2005.

[91] J. Davis, "Microvenus.pdf," *Art Journal,* vol. 55, pp. 70-74, 1996.

[92] H. Yokoo and T. Oshima, "Is Bacteriophage X174 DNA a Message from an Extraterrestrial Intelligence?," *Icarus,* vol. 38, pp. 148-153, 1979.

[93] M. R. Hennings and D. M. Kettelberger, "Genetic Music " Unted states of America Patent, 2004.

[94] M. B. Beck and R. V. Yampolskiy, "Hiding Color Images in DNA Sequences " presented at the 26th Modern Artificial Intelligence and Cognitive Science Conference (MAICS 2015), Greensboro, North Carolina, 2015.

[95] P. McClure, "Project 6: Genetic Algorithm with Wisdom of Artificial Crowds for Homophonic Substitution Ciphers," University of Louisville2012.

[96] M. B. Beck, A. H. Desoky, E. C. Rouchka, P. McClure, and R. V. Yampolskiy, "Using Genetic Algorithm and Wisdom of Artificial Crowds to Find Hidden Data in DNA," in *Embodying Intelligence in Multimedia Data Hiding*, ed: Science Gate Publishing 2015.

[97] S. Karlin and C. Burge, "Dinucleotide relative abundance extremes: a genomic signature," *Trends in Genetics,* vol. 11, pp. 283-290, 1995.

[98] C. Burge, A. M. Campbell, and S. Karlin, "Over- and under-representation of short oligonucleotides," *Proceedings National Academy of Science USA,* vol. 89, pp. 1358-1362, 1992.

[99] A. B. Stergachis, E. Haugen, A. Shafer, W. Fu, B. Vernot, and A. Reynolds, "Exonic Transcription Factor Binding Directs Codon Choice and Affects Protein Evolution," *Science 28 September 2012,* vol. 342, pp. 1367-1371, 2013.

APPENDIX A

Coding table for Improved Huffman coding scheme (text)

	Group 1 Code= G		**Group 2 Code= TT**		**Group 3 Code = TA**	
No.	**Character**	**DNA**	**Character**	**DNA**	**Character**	**DNA**
1	Space	AT	N	AT	.	AT
2	E	CT	S	CT	u	CT
3	shift	TC	H	TC	,	TC
4	T	TG	R	TG	w	TG
5	A	AC	D	AC	m	AC
6	O	AG	L	AG	f	AG
7	I	CG	C	CG	y	CG
8	G	AAT	3	AAT	;	AAT
9	P	AAC	4	AAC	q	AAC
10	B	AAG	5	AAG	z	AAG
11	V	CAT	6	CAT	<	CAT
12	-	CAA	7	CAA	=	CAA
13	(	CAC	8	CAC	%	CAC
14	)	CAG	9	CAG	+	CAG
15	K	CCA	J	CCA	*	CCA
16	0	CCT	X	CCT	?	CCT
17	1	CCG	/	CCG	>	CCG
18	2	CCC	:	CCC	Tab	CCC
19	Return	AAAT	$	AAAT	{	AAAT
20	^	AAAA	&	AAAA	}	AAAA
21	_	AAAC	~	AAAC	“	AAAC
22	#	AAAGC	[	AAAGC	\	AAAGC
23	@	AAAGT	]	AAAGT	|	AAAGT
24	!	GTCGCCG				
25	Page break	GTCTACCC				

APPENDIX B

Coding table for improved Huffman coding scheme (images)

DNA	**Symbol**
G	;
TT	,
TA	0
AT	1
CT	2
TC	3
TG	4
AC	5
AG	6
CG	7
AAT	8
AAC	9
AAG	S (s; x1; y1; a)
CAT	R (l; w; x1; y1; a)
CAA	L (x1; y1; x2; y2)
CAC	C (r; x1; y1)
CAG	P (n; x1; y1; x2; y2; x3; y3)
CCT	Tri
CCA	E

Shape	**Letter**	**Parameters**
Square	S	s-side units; x1, y1-coordinates of upper right vertex; a- angle of base
Rectangle	R	l- length units; w- width units; a-angle of base
Line	L	x1; y1; x2; y2 coordinates of line ends
Circle	C	r –radius; x1, y1 - coordinates of center
Polynom	P	n-order, xn; yn- parameters of points
Triangle	Tri	s1- side1; an – anangle; s2- side2; x1, y1 - coordinates of vertex a- angle of base
Ellipse	E	x1, y1 –center coordinates, l1- major axis; l2- minor axis

APPENDIX C

Coding table for improved Huffman coding scheme (music)

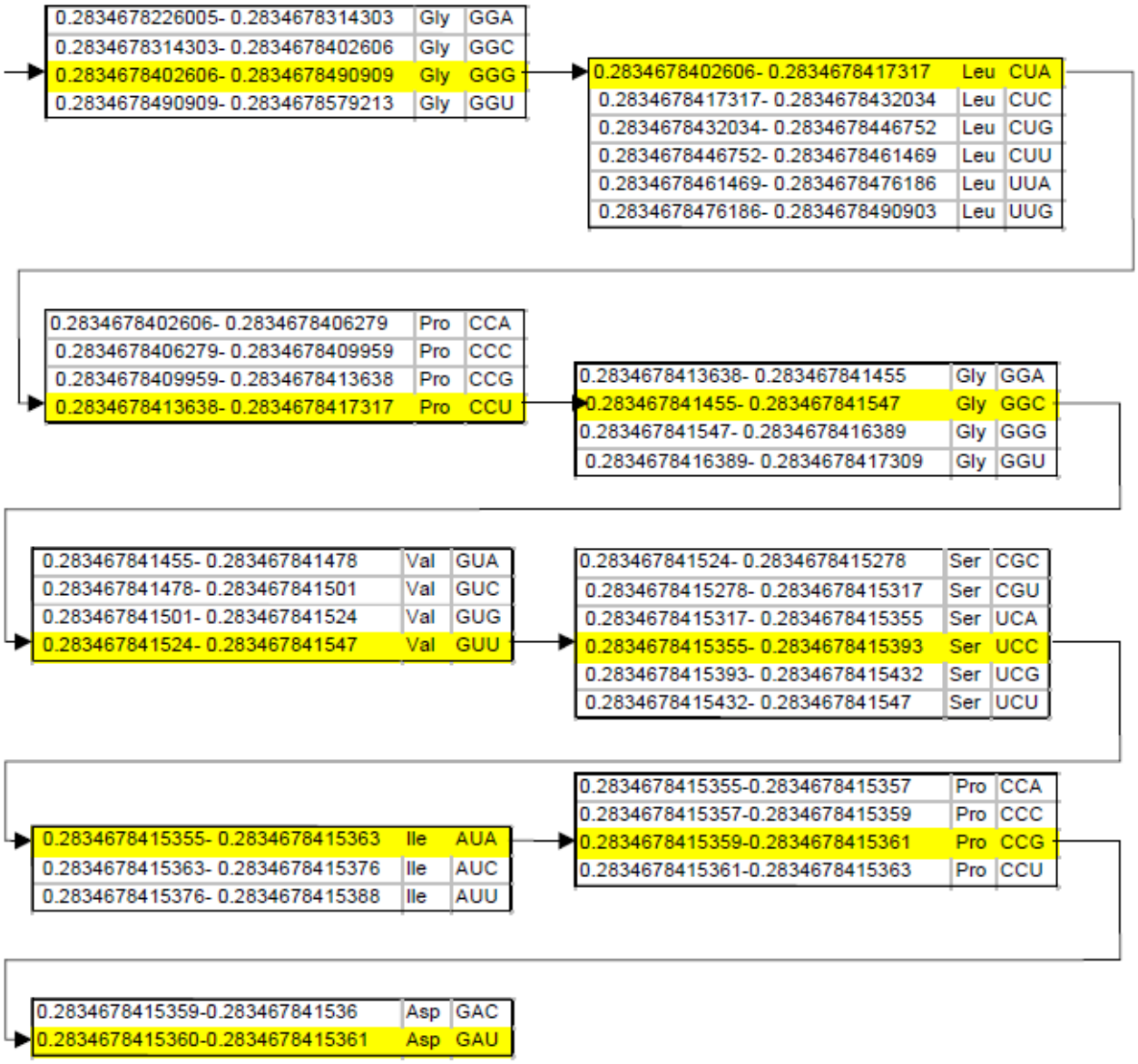
DNA	Music Note	Description
G	Quarter note (1/4)	Note Values
TT	Half note (1/2)	
TA	Whole note (1)	
AT	Eighth note (1/8)	
CT	Sixteenth note (1/16	
TC	Dot(.)	
AC	A	Note Pitches
AG	B	
CG	D	
AAT	E	
AAC	F	
AAG	G	
CAT	2/4 (meter)	Meter
CAA	3/4 (meter)	
CAC	4/4 (meter)	
CAG	(	Repeat
CCA	)	
CCT	X	
CCG	2	
CCC	3	
AAAT	4	

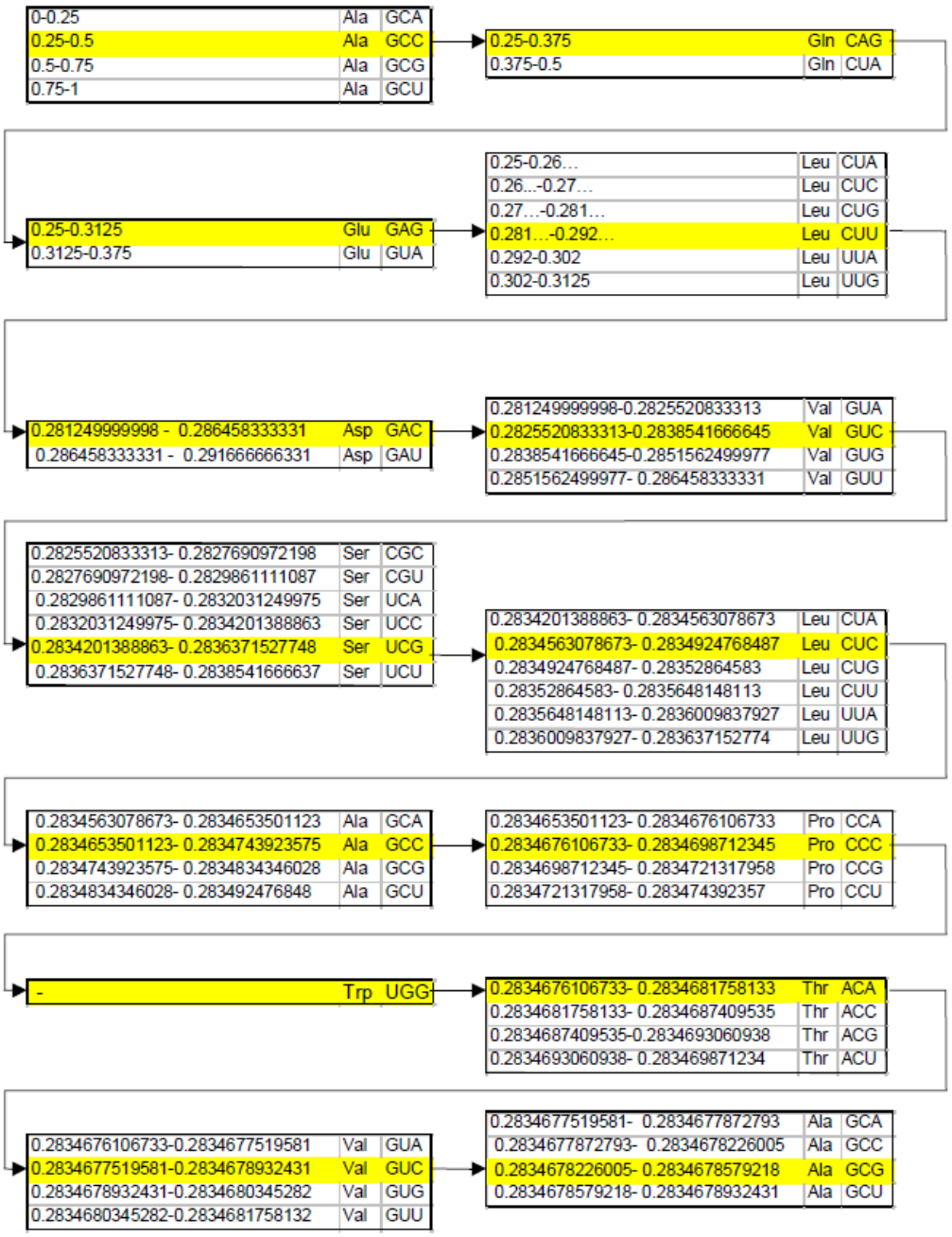
Ma- ry had a lit- tle lamb lit- tle lamb, lit- tle
m u * 4/4 (B1/4 A1/4 G1/4 A1/4 B1/4 B1/4 B1/2 A1/4 A1/4 A1/2 B1/4 D1/4
taac tact tacca cac cag ACG TGG AAGG TGG ACG ACG ACTT TGG TGG TGTT ACG CGG

lamb Ma- ry had a lit- tle lamb,its fleece was white as snow
D1/2 B1/4 A1/4 G1/4 A1/4 B1/4 B1/4 B1/4 B1/4 A1/4 A1/4 B1/4 A1/4 G1) x 2
CGTT ACG TGG AAGG TGG ACG ACG ACG ACG TGG TGG ACG TGG AAGTA cca cct ccg

APPENDIX D

Arithmetic Encoding




0-0.25 Ala GCA
0.25-0.5 Ala GCC
0.5-0.75 Ala GCG
0.75-1 Ala GCU
0.25-0.375 Gln CAG
0.375-0.5 Gln CUA
0.25-0.3125 Glu GAG
0.3125-0.375 Glu GUA
0.25-0.26… Leu CUA
0.26…-0.27… Leu CUC
0.27…-0.281… Leu CUG
0.281…-0.292… Leu CUU
0.292-0.302 Leu UUA
0.302-0.3125 Leu UUG
0.281249999998 - 0.286458333331 Asp GAC
0.286458333331 - 0.291666666331 Asp GAU
0.281249999998-0.2825520833313 Val GUA
0.2825520833313-0.2838541666645 Val GUC
0.2838541666645-0.2851562499977 Val GUG
0.2851562499977- 0.286458333331 Val GUU
0.2825520833313- 0.2827690972198 Ser CGC
0.2827690972198- 0.2829861111087 Ser CGU
0.2829861111087- 0.2832031249975 Ser UCA
0.2832031249975- 0.2834201388863 Ser UCC
0.2834201388863- 0.2836371527748 Ser UCG
0.2836371527748- 0.2838541666637 Ser UCU
0.2834201388863- 0.2834563078673 Leu CUA
0.2834563078673- 0.2834924768487 Leu CUC
0.2834924768487- 0.28352864583 Leu CUG
0.28352864583- 0.2835648148113 Leu CUU
0.2835648148113- 0.2836009837927 Leu UUA
0.2836009837927- 0.283637152774 Leu UUG
0.2834563078673- 0.2834653501123 Ala GCA
0.2834653501123- 0.2834743923575 Ala GCC
0.2834743923575- 0.2834834346028 Ala GCG
0.2834834346028- 0.283492476848 Ala GCU
0.2834653501123- 0.2834676106733 Pro CCA
0.2834676106733- 0.2834698712345 Pro CCC
0.2834698712345- 0.2834721317958 Pro CCG
0.2834721317958- 0.283474392357 Pro CCU
- Trp UGG
0.2834676106733- 0.2834681758133 Thr ACA
0.2834681758133- 0.2834687409535 Thr ACC
0.2834687409535-0.2834693060938 Thr ACG
0.2834693060938- 0.283469871234 Thr ACU
0.2834676106733-0.2834677519581 Val GUA
0.2834677519581-0.2834678932431 Val GUC
0.2834678932431-0.2834680345282 Val GUG
0.2834680345282-0.2834681758132 Val GUU
0.2834677519581- 0.2834677872793 Ala GCA
0.2834677872793- 0.2834678226005 Ala GCC
0.2834678226005- 0.2834678579218 Ala GCG
0.2834678579218- 0.2834678932431 Ala GCU

APPENDIX E

Coding table for 64 color bitmap

Codon	R	G	B	Color	Sample
AAA	0	0	0	Black	
AAC	0	0	84		
AAG	0	0	127		
AAT	0	0	255	Blue	
ACA	0	84	0		
ACC	0	84	84		
ACG	0	84	127		
ACT	0	84	255		
AGA	0	127	0		
AGC	0	127	84		
AGG	0	127	127		
AGT	0	127	255	Slate Blue	
ATA	0	255	0	Green	
ATC	0	255	84		
ATG	0	255	127	Spring Green	
ATT	0	255	255		
CAA	84	0	0		
CAC	84	0	84		
CAG	84	0	127		
CAT	84	0	255		
CCA	84	84	0		
CCC	84	84	84	Grey	
CCG	84	84	127		
CCT	84	84	255		
CGA	84	127	0		
CGC	84	127	84		
CGG	84	127	127		
CGT	84	127	255		
CTA	84	255	0		
CTC	84	255	84		
CTG	84	255	127		
CTT	84	255	255		
GAA	127	0	0		
GAC	127	0	84		
GAG	127	0	127		

GAT	127	0	255		
GCA	127	84	0		
GCC	127	84	84		
GCG	127	84	127		
GCT	127	84	255		
GGA	127	127	0		
GGC	127	127	84		
GGG	127	127	127		
GGT	127	127	255		
GTA	127	255	0	Chartreuse	
GTC	127	255	84		
GTG	127	255	127		
GTT	127	255	255		
TAA	255	0	0	Red	
TAC	255	0	84		
TAG	255	0	127		
TAT	255	0	255	Magenta	
TCA	255	84	0		
TCC	255	84	84		
TCG	255	84	127		
TCT	255	84	255		
TGA	255	127	0	Dark Orange1	
TGC	255	127	84		
TGG	255	127	127		
TGT	255	127	255		
TTA	255	255	0	Yellow	
TTC	255	255	84		
TTG	255	255	127		
TTT	255	255	255	White	

APPENDIX F

Unigrams, sorted by frequency:

[space]	16.27371%		"	0.10904189%
E	9.787157%		9	0.09536828%
T	7.224867%		`	0.001183685%
A	6.470553%		>	7.959262E-4%
O	6.083978%		<	4.2857564E-4%
I	6.0378346%		^	8.163345E-5%
N	5.815098%		D	2.9236817%
S	5.345216%		C	2.7640884%
R	5.0819883%		U	2.2520833%
H	3.4739728%		M	2.0308976%
L	3.3034813%		P	1.8397732%
F	1.7573234%		3	0.08010283%
G	1.580138%		5	0.07524563%
Y	1.3779523%		4	0.06742923%
W	1.3059107%		8	0.06073529%
B	1.2320325%		7	0.056816883%
.	0.9403153%		6	0.056510758%
V	0.88635564%		/	0.035612594%
,	0.8497226%		;	0.027673742%
K	0.5261276%		!	0.021490006%
?	0.30443156%		[	0.021163473%
0	0.24324727%		]	0.021041023%
-	0.21561435%		_	0.01908182%
1	0.20849183%		%	0.014163404%
X	0.18610387%		$	0.012326651%
2	0.14506264%		*	0.009040905%
J	0.13220537%		&	0.008734779%
'	0.1176134%		=	0.005510258%
)	0.11216437%		#	0.0018775694%
(	0.11083782%		+	0.0017143026%
Z	0.08887842%		\	0.0013877687%
Q	0.08361306%		@	0.0012449102%
:	0.08167427%			

APPENDIX G

Messages used for testing

I AGREE THAT THE AREA IS VERY LARGE. IT IS ALSO VERY GREEN. I HEAR THAT IT IS EAST OF THE TREE LINE AND SOUTH OF TOWN. MY PET LIKES TO SLEEP IN THE FIELD. I GO THERE TO FEED A VERY LARGE DUCK. THE TREE TOWER HAD A RED PATTERN.

SAVING INFLOWS FROM ABROAD CAN BE BENEFICIAL IF THE COUNTRY THAT RECEIVES THOSE INFLOWS INVESTS THEM WELL. UNFORTUNATELY, THAT WAS NOT ALWAYS THE CASE IN THE UNITED STATES AND SOME OTHER COUNTRIES. FINANCIAL INSTITUTIONS REACTED TO THE SURPLUS OF AVAILABLE FUNDS BY COMPETING AGGRESSIVELY FOR BORROWERS, AND, IN THE YEARS LEADING UP TO THE CRISIS, CREDIT TO BOTH HOUSEHOLDS AND BUSINESSES BECAME RELATIVELY CHEAP AND EASY TO OBTAIN. ONE IMPORTANT CONSEQUENCE WAS A HOUSING BOOM IN THE UNITED STATES, A BOOM THAT WAS FUELED IN LARGE PART BY A RAPID EXPANSION OF MORTGAGE LENDING. UNFORTUNATELY, MUCH OF THIS LENDING WAS POORLY DONE, INVOLVING, FOR EXAMPLE, LITTLE OR NO DOWN PAYMENT BY THE BORROWER OR INSUFFICIENT CONSIDERATION BY THE LENDER OF THE BORROWER'S ABILITY TO MAKE THE MONTHLY PAYMENTS. LENDERS MAY HAVE BECOME CARELESS BECAUSE THEY, LIKE MANY PEOPLE AT THE TIME, EXPECTED THAT HOUSE PRICES WOULD CONTINUE TO RISE--THEREBY ALLOWING BORROWERS TO BUILD UP EQUITY IN THEIR HOMES--AND THAT CREDIT WOULD REMAIN EASILY AVAILABLE, SO THAT BORROWERS WOULD BE ABLE TO REFINANCE IF NECESSARY. REGULATORS DID NOT DO ENOUGH TO PREVENT POOR LENDING, IN PART BECAUSE MANY OF THE WORST LOANS WERE MADE BY FIRMS SUBJECT TO LITTLE OR NO FEDERAL REGULATION.

CURRICULUM VITAE

Marc B. Beck

1022 South Brook Street
Apartment 5
Louisville, KY 40203

(270)-313-2386
Marcbeck1982@yahoo.com

PROFESSIONAL INTERESTS

My current research focuses on digital forensics and bioinformatics. I am proficient in several programming languages including HTML, C, C++, Visual Basic, Java, Python, C#, XNA and Matlab.

CITIZENSHIP

Germany

EDUCATION

PhD in Computer Engineering and Computer Science May 2015	**University of Louisville, Louisville/ Kentucky 40292** Dissertation Title: A Forensics Software Toolkit for DNA Steganalysis
Doctoral Committee:	Roman Yampolskiy (advisor), Ahmed Desoky (co-advisor), Eric Rouchka, Ibrahim Imam, John Naber
M.S. in Industrial Technology August 2009	Morehead State University, Morehead, Kentucky 40351 Thesis Title: Remote Control And Automation of VHF/UHF Satellite Tracking System
Thesis Committee:	Benjamin Malphrus, Ahmad Zargari, William Grise, Jeffrey Kruth
	Brescia University, Owensboro, Kentucky 42301
B.S. in Computer Science May 2007	

POSITIONS

January 2011-present University of Louisville, Louisville/ Kentucky
Research Assistant funded through CECS department at University of Louisville

August 2009-June 2010 University of Louisville, Louisville/ Kentucky
Research Assistant/Kentucky Space Initiative (KySat) funded through Kentucky Science & Technology Corporation (KSTC)

June 2008-July 2009 Morehead State University, Morehead/ Kentucky
Research Assistant/Kentucky Space Initiative (KySat) funded through Kentucky Science & Technology Corporation (KSTC)

August 2007-May 2008 Morehead State University, Morehead/ Kentucky
Graduate Assistant at Space Science Center
-wrote program to create images from astronomical data
-tested RAID array for data storage
-set up and tested KySat components, such as S-Band radio and development board.

July 2002- August 2003 Alternate Computers, Giessen/Germany
Service technician
-Tested and repaired PC hardware

RESEARCH TOPICS

2007-2008 Developed software to create images from radio telescope data

2008-2010 Satellite Tracking software and groundstation networks

2010-2011 Voice Recognition software

2011-present DNA forensic software

PUBLICATIONS

Beck, Marc B., Yampolskiy, Roman V. Hiding Color Images in DNA Sequences 26th Modern Artificial Intelligence and Cognitive Science Conference (MAICS 2015). April 25-26, 2015 Greensboro, North Carolina, USA. (in press).

Beck, Marc B., Rouchka, Eric C., Desoky, Ahmed H., McClure, Patrick S., Yampolskiy, Roman V. Using Genetic Algorithm and Wisdom of Artificial Crowds to Find Hidden Data in DNA. Embodying Intelligence in Multimedia Data Hiding. Science Gate Publishing (in press).

Beck, M. B., Desoky, A. H., Rouchka, E. C., & Yampolskiy, R. V. (2014). “Decoding Methods for DNA Steganalysis.” Paper presented at the 6th International Conference on Bioinformatics and Computational Biology (BICoB 2014) Las Vegas, Nevada, USA.

Marc B. Beck, Eric C. Rouchka, and Roman V. Yampolskiy, "Finding Data in DNA: Computer Forensic Investigations of Living Organisms." Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunication Engineering, vol. 114, pp. 204-219, 2013.

Beck, Marc B., Yampolskiy, Roman V. DNA as a Medium for Hiding Data. BMC Bioinformatics Volume 13 Issue Suppl 12 (2012)

M Boukhris, AA Mohamed, D D'Souza, M Beck, NE Ben Amara, RV Yampolskiy, "Artificial human face recognition via Daubechies wavelet transform and SVM",
16th International Conference on Computer Games (CGAMES), 2011, 18-25

S Krijestorac, M Beck, J Bagby, "Border Gateway Protocols", International Journal of Modern Engineering Spring/Summer 2011 Volume 11 Number2

AWARDS

2014	First Place at the Annual Kentucky Academy of Science meeting for Graduate Research Competition
2013	Third Place at the Annual Kentucky Academy of Science meeting for Graduate Research Competition
2009	Second Place at the Annual Kentucky Academy of Science meeting for Graduate Research Competition
2008	First Place at the Annual Kentucky Academy of Science meeting for Graduate Research Competition
2007	Second Place at the Annual Kentucky Academy of Science meeting for Graduate Research Competition
2005	joined honors program at Brescia University
2004-2007	mentions on Dean's List in five semesters for GPA greater than 3.5

HONOR SOCIETIES

- Alpha Chi (since 2006)
- Delta Epsilon Sigma (since 2007)

PROFESSIONAL MEMBERSHIP

Member ACM, IEEE

REFERENCES

Available upon request