San Jose State University
SJSU ScholarWorks

Master's Projects | Master's Theses and Graduate Research

2009

Heuristic Search Cryptanalysis of the Zodiac 340 Cipher

Pallavi Kanagalakatte Basavaraju
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the Computer Sciences Commons

Recommended Citation

Basavaraju, Pallavi Kanagalakatte, "Heuristic Search Cryptanalysis of the Zodiac 340 Cipher" (2009). *Master's Projects*. 56.
DOI: https://doi.org/10.31979/etd.4krr-6vhr
https://scholarworks.sjsu.edu/etd_projects/56

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

HEURISTIC SEARCH CRYPTANALYSIS OF THE ZODIAC 340 CIPHER

A Project Report

Presented to

The faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Computer Science

By

Pallavi Kanagalakatte Basavaraju

December 2009

© 2009

Pallavi Kanagalakatte Basavaraju

ALL RIGHTS RESERVED

SAN JOSÉ STATE UNIVERSITY

The Undersigned Project Committee Approves the Project Titled

HEURISTIC SEARCH CRYPTANALYSIS OF THE ZODIAC 340 CIPHER

by
Pallavi Kanagalakatte Basavaraju

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Mark Stamp	Department of Computer Science	Date
Dr. Sami Khuri	Department of Computer Science	Date
Mr. Ashish Dobhal	Cadence Design Systems	Date

APPROVED FOR THE UNIVERSITY

Associate Dean	Office of Graduate Studies and Research	Date

ABSTRACT

HEURISTIC SEARCH CRYPTANALYSIS OF THE ZODIAC 340 CIPHER

by Pallavi Kanagalakatte Basavaraju

The Zodiac 340 cipher is one of the most famous unsolved ciphers of all time. It was allegedly written by “the Zodiac”, whose identity remains unknown to date. The Zodiac was a serial killer who killed a number of people in and around the San Francisco Bay area during the 1960s. He is confirmed to have seven victims, two of whom survived [1], although in taunting letters to the news media he claims to have killed 37 people. During this time, an encrypted message known as the Zodiac 408 cipher was mailed to 3 different newspapers in the San Francisco bay area. This was a homophonic cipher and was successfully decoded. Within a few days he sent out another cipher that was 340 characters long [4]. This cipher, which is known as the Zodiac 340 cipher, is unsolved to date. Many cryptologists have tried to crack this cipher but with no success.

In this project, we implemented a novel genetic algorithm in an attempt to crack the Zodiac 340 cipher. We have attacked the cipher as a homophonic cipher where each cipher symbol is mapped to only a single English letter, but each English letter can be mapped to multiple cipher symbols. In the genetic algorithm, we implemented two variants of crossover: simple and intelligent. The simple crossover looks for commonly occurring substrings, without looking for actual English words in a putative decrypt. The intelligent crossover counts the number of actual English words that can be found in a putative decrypt when evaluating each solution. We implemented a dictionary lookup for quickly identifying English words for the intelligent crossover. The genetic algorithm using a combination of simple and intelligent crossovers was able to identify many English words in various putative decrypts but no solution was found.

ACKNOWLEDGEMENTS

I express my sincere thanks to my advisor, Dr Mark Stamp, for his guidance, support and patience, without whom my research study would not have succeeded. I would also like to thank Dr Sami Khuri and Ashish Dobhal for their valuable suggestions and useful comments.

Furthermore, I would like to thank my caring husband, Anup Hosangadi, and my parents for their support, patience and encouragement.

TABLE OF CONTENTS

1. Introduction 1
2. Zodiac Ciphers 2
3. Zodiac 340 cipher 6
3.1 Comparison of the Z340 and the Z408 ciphers 7
3.2 Symbol frequencies of Z340, Z408 and English 8
3.3 Study of encryption techniques 9
3.4 Previous attempts at decoding the Z340 cipher. 11
4. Algorithms 13
4.1 Simulated annealing 13
4.2 Ant colony optimization 13
4.3 Genetic algorithm 14
4.3.1 Evolutionary processes in Biology 14
4.3.2 Roulette wheel method 15
5. Analyzing the Z340 using a genetic algorithm 16
5.1 Creating frequency statistics of English alphabets 16
5.2 Frequency statistics for symbols in the Z340 16
5.3 Creating initial solutions 16
5.4 Scoring putative solutions 18
5.4.1 Graph scores 18
5.4.2 Scoring using dictionary lookup 19
5.5. Crossover using Roulette method 19
5.6 Crossover 21
5.6.1 Simple crossover 21
5.6.2 Intelligent crossover 23
5.6.2.1 Ternary search tree (TST) 23
5.6.2.2 Intelligent crossover technique 29
5.6.2.3 Intelligent word score (Score normalization) 30
5.7 Mutation 32
5.8 Experimental results 32
5.8.1 Different flows of the genetic algorithm 32

5.8.2 Testing genetic algorithm on the Z408 cipher 33
5.8.3 Results for the Z340 cipher 35
6. Future work 39
7. Conclusion 39
Appendix A: References 40
Appendix B: Interesting decrypts found for Z340 cipher 42
Appendix C: Interesting decrypts found for Z408 cipher 44
Appendix D: Zodiac cover letters 45
a) Zodiac 408 cover letter 45
b) Zodiac 340 cover letter 48
c) Z13 / My Name Is …. Cover letter and Bomb Diagram 49
d) Z32 cover letter 51

LIST OF TABLES, FIGURES AND EQUATIONS

Tables:

Table 1: Z340 in a numeric form 6

Table 2: Entropy comparison 9

Table 3: Experiments on Z408 cipher 33

Table 4: Experiments on Z340 cipher without using cribs 35

Table 5: Experiments on Z340 cipher using crib (HER, ZODIAC) 36

Table 6: Experiments on Z340 cipher using crib (KILL) 37

Figures:

Figure 1: Z408 part 1 2

Figure 2: Z408 part 2 3

Figure 3: Z408 part 3 3

Figure 4: Z340 cipher 4

Figure 5: Z13, Zodiac's “My name is ...” cipher 5

Figure 6: Z32, Zodiac's “Button” cipher 5

Figure 7: Z340 cipher symbol occurrences 7

Figure 8: Z408 cipher symbol occurrences 7

Figure 9: English letter frequencies 8

Figure 10: Columnar transposition cipher 10

Figure 11: Keyword columnar transposition cipher 10

Figure 12: Example of crossover operation 15

Figure 13: Algorithm to create initial solutions 17

Figure 14: Pie-chart representation of solutions 20

Figure 15: Flow chart for simple crossover (producing child C1 from parent P1 and P2) 22

Figure 16: Ternary tree 24

Figure 17: Algorithm to insert word in the dictionary 24

Figure 18: Representation of words in the dictionary 25

Figure 19: Algorithm to find word in the dictionary 27

Figure 20: Flow chart for intelligent crossover (producing child C1 from parent P1 and P2) 29

Figure 21: Flowchart for genetic algorithms 32

Figure 22: Z408 part 1, July 31, 1969 Times Herald letter 45

Figure 23: Z408 part 2, July 31, 1969 Chronicle letter 46

Figure 24: Z408 part 3, July 31, 1969 Examiner letter 47

Figure 25: Z340 cover letter 48

Figure 26: Z13 cover letter 49

Figure 27: Z13 Bomb Diagram 50

Figure 28: Z32 button cover letter 51

Equations:

Equation 1: Shannon's entropy equation 8

Equation 2: Equation to calculate the solution score 19

Equation 3: Equation to calculate intelligent word score 31

Equation 4: The formula to calculate the word score 31

Equation 5: The formula to calculate the graph score 31

1. Introduction

Cryptography is an important subject in the modern digital age. It is a branch of mathematics and computer science that deals with securing messages. The sender encrypts the message using an algorithm and a key, and the receiver uses a key and another algorithm to decrypt the message. A good encryption algorithm is designed so that it is computationally infeasible for someone other than the receiver to decode the message.

Some famous modern cryptographic algorithms are RSA and DES. In the digital age, securely transmitting sensitive information has become an important problem and much work has been done to develop effective and fast algorithms for the encryption and decryption of messages. Some common applications of cryptography are in ATM machines, email and VPNs (Virtual Private Networks) [22].

Unfortunately, cryptography can be used by criminals to safely communicate messages without the law gaining knowledge of their activities. The police and FBI employ cryptography experts who attempt to decode such messages. One of the most famous unsolved ciphers is the “Zodiac 340 cipher” that was used by a serial killer to communicate with the press. The Zodiac operated in Northern California during the late 1960’s. His identity remains unknown to date.

Former FBI profiler John Douglas stated that many serial killers are motivated by a "desire to create and sustain their own mythology" [16]. The Zodiac Killer is one such murderer who left traces of his criminal activity. These traces were in the form of letters to the press, some of which included ciphers. The first of the ciphers that Zodiac created was the Zodiac 408 cipher, which was first sent to a local newspaper with a cover letter stating that it was from the killer. This cipher was decrypted and the message in that cipher described the intentions and motivations of the killer.

Many ciphers were sent out by the Zodiac after the Zodiac 408 cipher. The largest of the unsolved Zodiac ciphers is the Zodiac 340 cipher. The Zodiac 340 cipher is a 340 character long cipher that Zodiac mailed to the San Francisco Chronicle. The main aim of this project is to try to decode the Zodiac 340 cipher by using genetic algorithms.

This paper is organized as follows: Section 2 gives a more detailed description of the Zodiac and his ciphers. Section 3 focuses on the Zodiac 340 cipher and some of the prior attempts at solving it. Section 4 explains some heuristic algorithms including simulated annealing and ant colony optimizations, while section 5 covers genetic algorithms in detail, and explains how it can be used to attack the Zodiac 340 cipher. Experimental results are presented at the end of section 5.

2. Zodiac Ciphers

There have been a number of encrypted messages sent to California newspapers in the 1960s that are attributed to the Zodiac killer. Only one cipher, the Zodiac 408 was successfully decoded. This section gives details about the different ciphers that are attributed to Zodiac [1].

First Cipher: Zodiac 408 (Z408)

The Zodiac killer sent out his first cipher on July 31, 1969. The cipher was divided into three parts and each part was mailed to a different newspaper. The first cipher was mailed to the Vallejo Times Herald, the second was mailed to the San Francisco Chronicle, and the third was mailed to the San Francisco Examiner. These three parts were then published in the respective local newspapers. Donald and Bettye Harden, residents of Salinas California were able to decrypt the Z408 cipher [1][3].

Part 1, sent to Vallejo Times-Herald

Figure 1: Z408 part 1

This cipher was decoded to yield the following:

"*I like killing people because it is so much fun. It is more fun than killing wild game in the forrest because man is the most dangerous animal of all To kill something gi..*"

Part 2, sent to San Francisco Chronicle

Figure 2: Z408 part 2

The second part was successfully decoded to yield the following:
"..ves me the most thrilling experence It is even better than getting your rocks off with a girl The best part of it is that when I die I will be reborn in paradice and all th.."

Part 3, sent to San Francisco Examiner

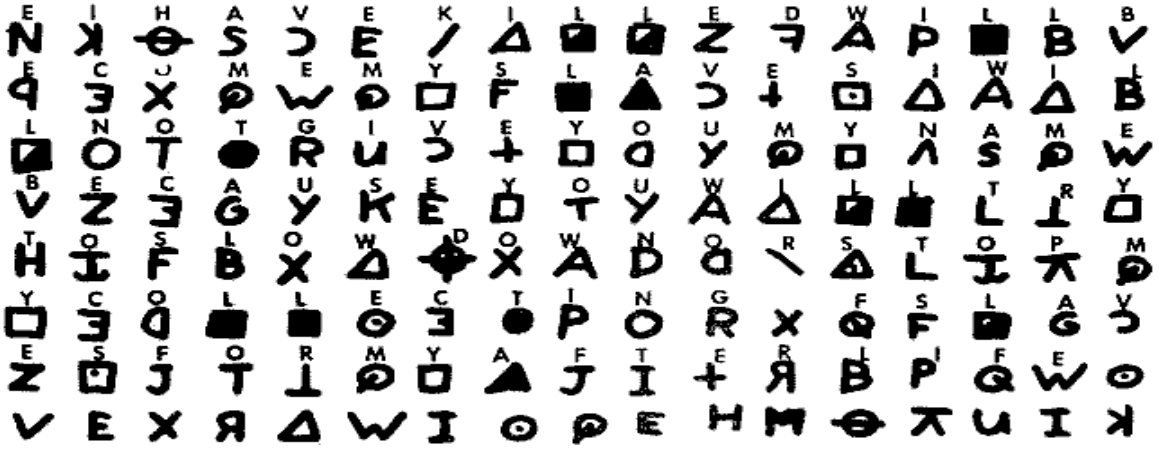


Figure 3: Z408 part 3

The third and final part was then decoded as:
"*.e I have killed will become my slaves I will not give you my name because you will try to slow down or stop my collecting of slaves for my afterlife*. "

The Z408 cipher is an example of a homophonic cipher. A homophonic cipher is defined as a one to many cipher, where each English alphabet is mapped to multiple symbols, but each symbol is mapped to only a single English letter. This cipher was manually decoded by the Hardens, without the use of any automated techniques.

Second Cipher: Zodiac 340 (Z340)

The next cipher was received at the San Francisco Chronicle on November 8, 1969. The cipher had 340 cipher letters and hence called Z340. The Z340 cipher is shown in its original form in Figure 4. Code-breakers have made various attempts to crack the Z340 cipher and obtain some meaningful results, but have failed to do so. Some of the methods that have been considered to decode the Z340 cipher include brute force method, dictionary based attacks and Hill-climb algorithm. These techniques are briefly discussed in section 3.4. As previously mentioned, the goal of this project is to use genetic algorithms to attempt to decode this cipher.

Z340 sent to San Francisco chronicle

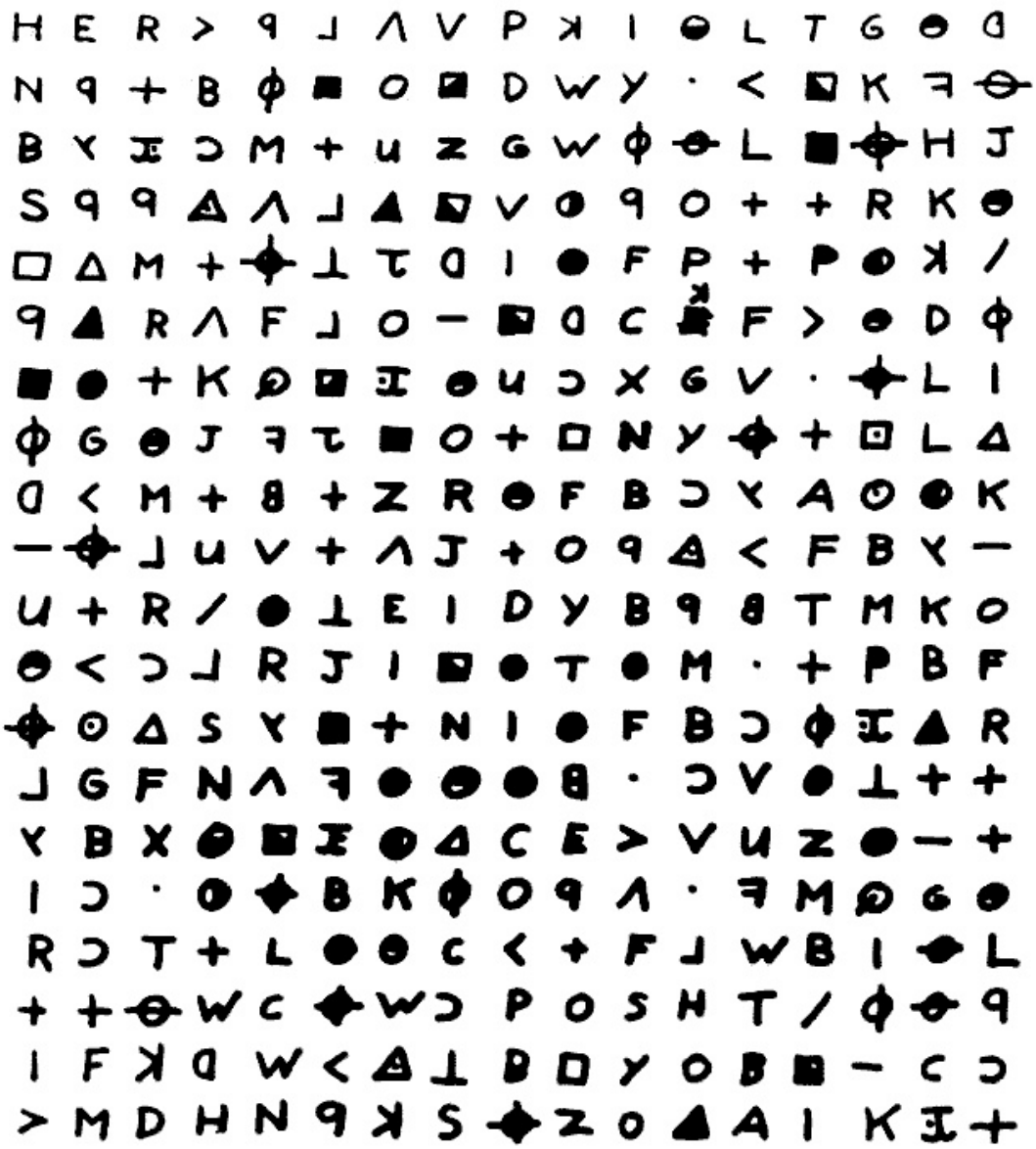


Figure 4: Z340 cipher

Third cipher: Z13

On April 20th, 1970 the Zodiac sent a letter to the San Francisco Chronicle, beginning with the sentence "This is the Zodiac speaking" [1]. The complete letter is available in Appendix D. In the letter, he appears to reveal his name using a cipher of 13 characters. This cipher is shown in Figure 5. This cipher is considered to be the third cipher sent by the Zodiac and is named as Z13 because of its length. Due to its brevity, the Z13 is a very hard cipher to decode. Longer cipher text messages have more data available for decryption methods to attempt a solution. Small ciphers are virtually impossible to decrypt.

Mr. Christopher Farmer, using a numerological approach had hinted at some potential clues such as "A Train 8 Blood 13" [7]. The key to obtaining this "solution" was the killer's operational name "ZODIAC". Other solutions that he hinted at are "A Train H Blood M", "813 Mt. Diablo Blvd" and "Mt. Diablo CT Street". However none of these putative solutions have been widely accepted [6].

Z13, sent to San Francisco Chronicle

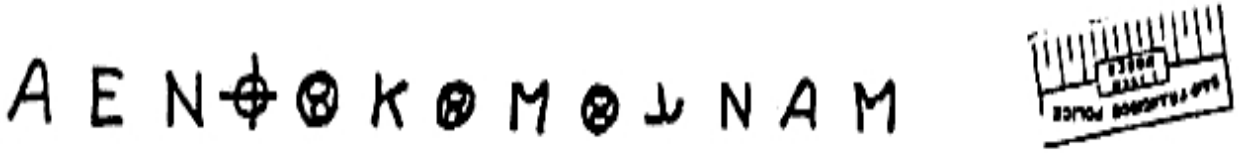


Figure 5: Z13, Zodiac's "My name is ..." cipher

Fourth cipher: Z32

Two months later on June 26th, 1970, the Zodiac sent his last cipher letter containing a cipher. This letter was sent to the San Francisco Chronicle [1]. In this letter, he expressed his displeasure about people not wearing Zodiac buttons as per his demands. He claimed to have killed a man sitting in a parked car with a .38 caliber handgun. Investigators initially believed that the person killed was Sgt Richard Reid who was shot with a .38 handgun a week before the letter [13]. But this theory was ultimately rejected because the Zodiac had used a 9mm handgun in his other crimes [17]. This is one of several examples where the Zodiac tried to take "credit" for murders that investigators believe he did not commit.

In this note, the Zodiac also threatened to blow up a school bus if his wishes were not met. He provided a map along with a cipher that he said indicated where the bomb was set. This cipher is 32 characters long and is known as the Z32 cipher and is shown in Figure 6. The entire letter is available in Appendix D.

Figure 6: Z32, Zodiac's "Button" cipher

3. Zodiac 340 cipher

The Z340 cipher has 340 characters consisting of 63 different symbols. In this project, we have translated the symbols to a numeric form, where each unique symbol is assigned a unique number. The Z340 cipher translated into the numeric form is shown below in Table 1. In the translated form, the number 1 in row 1 corresponds to the first symbol “H” in the cipher. Similarly, the number 2 corresponds to the symbol “E” and the number 3 and 4 corresponds to the symbols “R” and “>”, respectively, and so on.

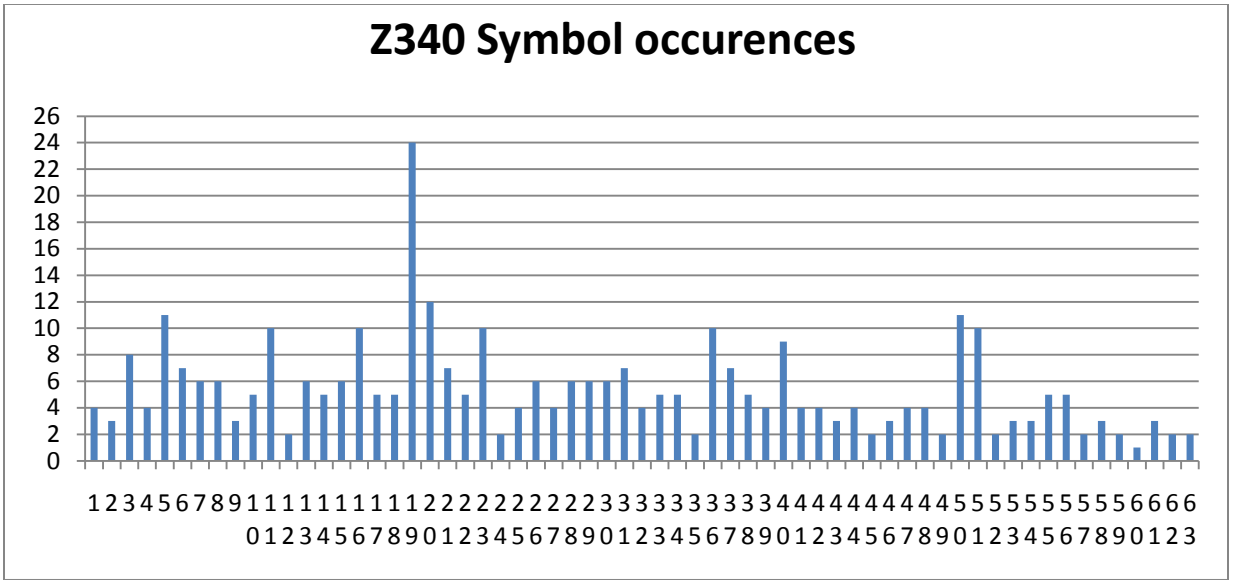
Table 1: Z340 in a numeric form

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	5	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
20	34	35	36	37	19	38	39	15	26	21	33	13	22	40	1	41
42	5	5	43	7	6	44	30	8	45	5	23	19	19	3	31	16
46	47	37	19	40	48	49	17	11	50	51	9	19	52	53	10	54
5	44	3	7	51	6	23	55	30	17	56	10	51	4	16	25	21
22	50	19	31	57	24	58	16	38	36	59	15	8	28	40	13	11
21	15	16	41	32	49	22	23	19	46	18	27	40	19	60	13	47
17	29	37	19	61	19	39	3	16	51	20	36	34	62	63	53	31
55	40	6	38	8	19	7	41	19	23	5	43	29	51	20	34	55
38	19	3	54	50	48	2	11	25	27	20	5	61	14	37	31	23
16	29	36	6	3	41	11	30	50	14	50	37	28	19	52	20	51
40	63	47	42	34	33	19	18	11	50	51	20	36	21	58	44	3
6	15	51	18	7	32	50	16	50	61	28	36	8	50	48	19	19
34	20	59	12	30	35	53	47	56	2	4	8	38	39	50	55	19
11	36	28	45	40	20	31	21	23	5	7	28	32	37	57	15	16
3	36	14	19	13	50	16	56	29	19	51	6	26	20	11	33	13
19	19	33	26	56	40	26	36	9	23	42	1	14	54	21	33	5
11	51	10	17	26	29	43	48	20	46	27	23	20	30	55	56	36
4	37	25	1	18	5	10	42	40	39	23	44	62	11	31	58	19

The total number of occurrences of each of the different symbol is shown in the chart in Figure 7. The X-axis represents the symbol number and the Y-axis represents the number of times the symbol appears in the cipher.

3.1 Comparison of the Z340 and the Z408 ciphers

The Z408 cipher (see section 2) uses 53 symbols and has a total of 408 characters. The number of occurrences of each symbol is shown in the chart in Figure 8. The Z408 cipher has a few symbols that appear a large number of times. The symbols 1, 2 and 26 appear 15, 21 and 16 times respectively. On the other hand in the Z340 cipher, the symbol 19 appears 24 times. This is twice the frequency of the second most frequently occurring symbol. Overall, the frequency distribution of the Z340 cipher is somewhat more uniform than the Z408 cipher.



63 different symbols →

Figure 7: Z340 cipher symbol occurrences

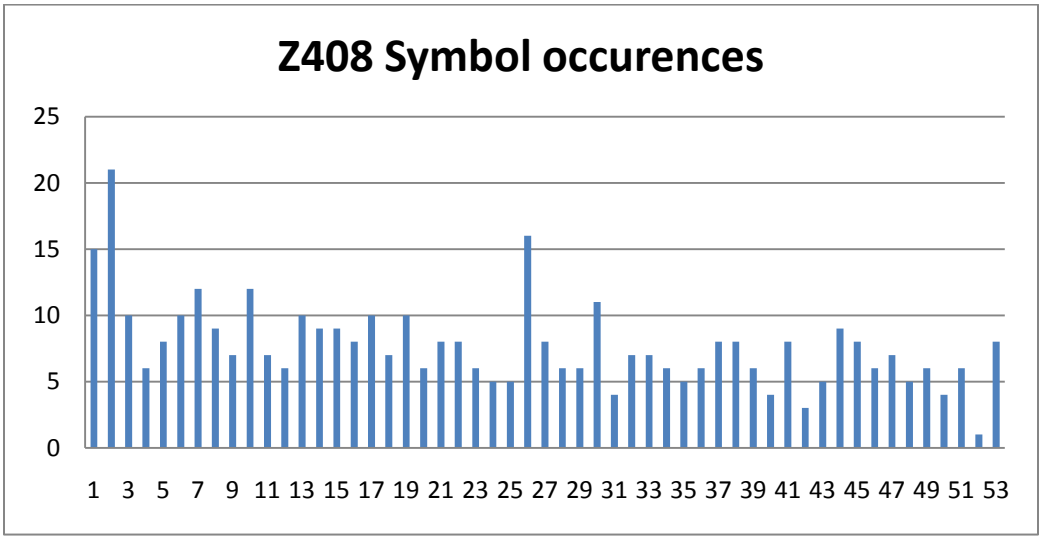


Figure 8: Z408 cipher symbol occurrences

3.2 Symbol frequencies of Z340, Z408 and English

The letter frequency for English text is shown in Figure 9 [15].

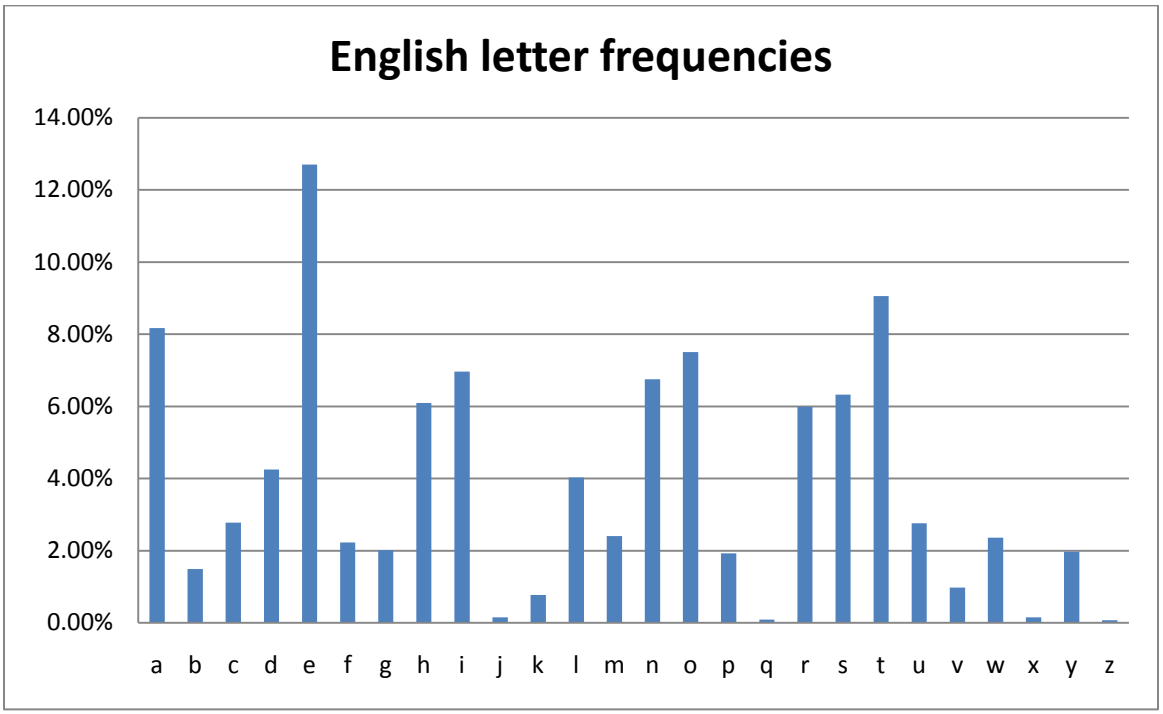


Figure 9: English letter frequencies

From Figure 9 it is evident that the English letter frequencies are not evenly distributed. The frequencies of some letters are a lot more than others. For example, the letter ‘e’ appears a lot more times than the letter ‘y’. However, we do not see such large variations in the symbol frequencies of the Z408 and Z340 ciphers. Both the ciphers use a lot more symbols than English. The symbol frequencies in these ciphers are more evenly balanced, except for a few symbols. This is done deliberately by Zodiac to make it difficult to decode using frequency based techniques. Since the frequencies of the symbols in the ciphers are more balanced than the English letters, it is evident that the ciphers cannot be simple substitution based ciphers. The Z408 was proven to be a homophonic cipher.

The frequencies of the Z408 and Z340 ciphers and the English letter frequencies can be further compared using Shannon’s entropy. Shannon’s entropy is widely used to quantify uncertainty [21]. Entropy in terms of number of bits is computed using Equation 1, shown below.

$$H(x) = -\sum_{x \in X} P(x) * Log(P(x))$$

Equation 1: Shannon’s entropy equation

The logarithm is to the base 2 and 0*log (0) is taken to be 0. Consider the letters in the English language. Assuming the probability of each letter is the same, the probability p(x) in the Entropy equation would be (1/26). This would be the probability for each letter in the English alphabet. Using Equation 1, the entropy is calculated as:

$$H(x) = -\sum\left(\frac{1}{26}\right) * \log\left(\frac{1}{26}\right) = \log(26) \approx 4.7$$

This implies that each letter in an English text would contain 4.7 bits of information. In actual English texts, the probability p(x) of each letter is not (1/26), as different letters have different frequencies (see Figure 9). For example, probability of letter 'a' is 0.08167. Using probabilities of English letters from their frequency statistics, the entropy of the English letters is computed to be 4.175. The entropies of the Z408 cipher symbols and the Z340 cipher symbols are also calculated using the frequency statistics shown in Figure 8 and Figure 7 respectively. The entropies calculated are shown in Table 2 below.

Table 2: Entropy comparison

English letters	4.175
Z340 cipher symbols	5.72
Z408 cipher symbols	5.68

Both the Z340 cipher and the Z408 cipher have higher entropies than English alphabets. But the entropy of Z340 is very similar to the entropy of the Z408 cipher. Though not a concrete proof, this indicates that the Z340 and Z408 ciphers may use similar encryption techniques. It is likely that the Z340 is also a homophonic cipher like the Z408. Since there are 63 unique symbols in the Z340 cipher and there are 26 English alphabets, the upper bound on the key space is 26^{63}. The key space represents the set of all possible keys for the cipher.

3.3 Study of encryption techniques

This section studies some of the common encryption techniques that could have been used to write the Z340 cipher. Modern encryption algorithms like RSA and Data Encryption Standard (DES) are not mentioned here because they were not available at the time Zodiac wrote his ciphers. .

(1) Homophonic substitution

In homophonic substitution, each English letter can be mapped to multiple cipher symbols, but each cipher symbol is mapped to a single English letter [11]. The Z408 cipher, which was successfully decoded, was written using homophonic substitution. In this project, we assume the Z340 cipher to be a homophonic cipher.

(2) Poly-alphabetic substitution

A poly-alphabetic cipher can be explained as a many to many cipher. Unlike the homophonic cipher, each symbol can be mapped to many English letters. As a result the key space of the poly-alphabetic cipher is very large. For the 340 character cipher in Z340, each character can be mapped to any of the 26 English alphabets. As a result, the upper bound on the key space is 26^{340}. The very large key space makes the problem very difficult to solve.

(3) Transposition ciphers

Transposition involves jumbling of the plain text in a manner that is unreadable by the attackers but can be read back by the intended person [5]. In the simple columnar transposition, a plain text is put into a matrix form with the key being the number of columns. For example consider the plain text KANAGALAKATT with key = 4. The text is transformed into a matrix with 4 columns as shown in Figure 10. The encrypted cipher is obtained by writing out the letters in each column, starting with the first column. In this example, the encrypted cipher text will be: KGKAAANLTAAT.

K	A	N	A
G	A	L	A
K	A	T	T

Figure 10: Columnar transposition cipher

The keyword columnar transposition is another form of the columnar transposition, where the order in which the columns are written out depends on the alphabetical order of the letters in a keyword. The column corresponding to the first letter (according to the alphabetical order) is written first, the column corresponding to the second letter is written next, and so on. For the previous example in Figure 10, consider the keyword PALU. Each letter in the keyword corresponds to a column in the matrix as shown in Figure 11. The letter ‘A’ is the first letter in the keyword (according to alphabetical order) and the column corresponding to this letter is written out first. The columns corresponding to the letters ‘L’, ‘P’ and ‘U’ in that order are then written out. The encrypted cipher text obtained is: AAANLTKGKAAT

P	A	L	U
K	A	N	A
G	A	L	A
K	A	T	T

Figure 11: Keyword columnar transposition cipher

In the double transposition cipher, both the rows and columns are interchanged. This is a very difficult cipher to decode.

(4) One-time pad

One time pads are the only real world ciphers that are provably secure [5]. In these ciphers, the plain text P is first converted to binary format and then a binary key K is generated whose length is the same as that of the plain text. The cipher text is now generated by performing a simple XOR operation on the plain text, $C = P \oplus K$
The plain text is retrieved by performing the same XOR operation on the cipher text along with the key K
$C \oplus K = (P \oplus K) \oplus K = P$

Example: suppose P = 01010111 and K = 10111101 then
$C = P \oplus K = 01010111 \oplus 10111101 = 11101010$
P can also be recovered
$P = C \oplus K = 11101010 \oplus 10111101 = 01010111$

3.4 Previous attempts at decoding the Z340 cipher.

This section describes some of the prior techniques that have been attempted to find a putative decrypt for the Z340 cipher.

(1) Using brute force method [4]

Brute force methods involve trying out all possible combinations to find a solution. Though they are guaranteed to find a solution if it exists, they are computationally infeasible for large problems. To decrypt the Z340 cipher, the key space is 63^{26}, assuming that it is a homophonic cipher. This number is approximately equal to $6.065*10^{46}$. Analyzing all possible keys for the Z340 cipher is therefore computationally infeasible.

(2) Using dictionary based attacks [23]

Dictionary based attacks use a brute force technique where all the words in an exhaustive list (dictionary) are successively tried [24]. The set of keys to be tried depends on the size of the word list and is generally much smaller than the key space of plain brute force method. Some techniques can be used to reduce the search space of the dictionary attack, such as finding patterns in the cipher text. For example, consider a portion of the cipher text as "ΛΦΩΦΛ". Based on the pattern in this text, and assuming that the cipher is homophonic, some of the English words that can be attempted are MADAM, LEVEL and METER.

In [23], the author developed an algorithm to decode homophonic ciphers by using a genetic algorithm guided by dictionary attack. This algorithm was tried on the Z408 cipher, which has a known decrypt. The technique was actually tested on a small portion of the cipher, consisting of 52 characters. The different solutions to the genetic algorithm were formed by guessing words from a short word list. The technique was able to correctly decode the selected cipher text. The good result of this technique can be attributed to: (i) The small size (52 characters) of the cipher

text selected, and (ii) The word list used for dictionary attack consisted of words from the known decrypt. The author proposed using this technique for finding a decrypt for the Z340 cipher. But his algorithm did not really prove how it can be used on a large cipher, having a large number of possible words.

In general, there are some disadvantages in using the dictionary based attacks to find a decrypt for the Z340 cipher. Firstly, the Z340 cipher uses a large number of symbols, which makes it difficult to find patterns. Secondly, the Z340 does not have a one-to-one mapping between the English letters and the cipher symbols. Most probably it has at least a one-to-many (English letter to symbols) mapping. This makes it harder for the dictionary based attacks to find a solution. Another problem with dictionary based attacks is that it will not work when the words in the cipher are misspelt.

(3) The Hill-climb algorithm has been used to find a putative decrypt for the Z340 cipher [12]. It is an improvement over the basic greedy search algorithm. The Hill-climb algorithm climbs multiple hills to find local optima. It then records the best solution among all the local optima that it has seen. The work in [12] used graph based scores (explained in section 5.4.1) to evaluate solutions. It found results with a few animal names but it was not able to derive any meaningful decrypts. In our experiments we have found that using only graph based scores tends to generate decrypts that have a lot of repeated words of small length. We use a combination of word scores (obtained from dictionary lookup) and graph scores in our genetic algorithm.

All of the above methods have failed to successfully decode the Z340 cipher. Furthermore none of them have been able to derive any likely plaintext message. This has led people to speculate that the Z340 may be a completely meaningless message.

In this project, we use a novel approach to analyze the Z340 cipher using genetic algorithms. A genetic algorithm tries to improve solutions by combining previous solutions. Using this approach it is possible to improve decoding of two solutions by combining their best parts.

4. Algorithms

This section gives a brief description of heuristic evolutionary algorithms. These algorithms have been used to search for solutions for several problems in engineering and sciences with large search space. Genetic algorithms are explained in section 4.3.

4.1 Simulated annealing

Simulated annealing is a global optimization heuristic that is inspired by the annealing process in metallurgy [2]. The annealing process in metallurgy deals with heating and controlled cooling of metals to increase the crystal size and reduce the defects. Simulated annealing technique begins with an initial solution and navigates through different solutions in search of an optimal solution.

The simulated annealing algorithm is guided by a temperature schedule. The temperature is high in the beginning and is decreased by a small amount during each iteration. Given a solution X, a neighboring solution Y is examined and if Y is a better solution (lower cost) than X, solution Y is accepted. Even if solution Y is not a better solution (higher cost) it might still be accepted based on a probability function, which is a function of the temperature. This probability is higher in the initial stages, when the temperature is higher. In the later stages, when the temperature decreases, the probability that a higher cost solution is accepted becomes lower.

Simulated annealing helps the search to come out of local minima by accepting some higher cost solutions. Simulated annealing algorithms have been successfully applied in solving difficult problems in VLSI physical design automation such as partitioning and global placement [25].

4.2 Ant colony optimization

The ant colony optimization algorithm is a probabilistic algorithm that can be used to find solutions to combinatorial problems that are reducible to finding good paths on graphs [9]. The algorithm is influenced by how ant colonies behave in the real world. Due to lack of vision, ants use pheromone trails to help them communicate information among themselves. After finding the food source, the ants find their way back to the colony leaving behind a trail of pheromone. Pheromone has a strong scent attached to it which helps the other ants to stop wandering around and find their way to the trail. This pheromone evaporates with time. When one ant finds a route with the shortest path, other ants are likely to use this path. With positive feedback, this path becomes the one with the strongest scent. The scent on the other longer paths begins to fade away. Finally all the ants end up using the shortest path.

The traveling salesman problem (TSP) was one of the first problem that was successfully solved by using the ant colony algorithm and the results showed that better results were achieved using the ant colony algorithm as compared to the traditional approach to the TSP [9].

4.3 Genetic algorithm

Genetic algorithm is a heuristic search technique that is used to find exact or approximate solutions to complex problems [2]. Genetic algorithm belongs to the class of evolutionary algorithms such as simulated annealing and ant colony optimizations.

Genetic algorithms are based on the natural processes in evolutionary biology such as inheritance, mutation and crossover. Genetic algorithms are used to find solutions to complex optimization problems in various fields such as in bioinformatics, chemistry, robotics, economics, physics, etc [20]. Examples of applications include scheduling algorithms, RNA sequence prediction in bioinformatics and code breaking in cryptography.

4.3.1 Evolutionary processes in Biology

Genes are the basic units of heredity in living organisms. A gene contains rules that describe how the organism is built up from the basic cells. Each gene contains information about specific traits of the organism such as height, skin color, intelligence, etc. They are connected into long strings called chromosomes. Genes are carried over from parents to their offspring. Occasionally, some genes may be mutated and it might appear as a totally new trait in the offspring. Mutation plays a very important role in the process of natural selection. According to the theory of evolution, organisms evolve over time to better adapt to the changes in the surrounding environment. Genetic algorithms mimic these processes in nature to solve complex problems. They are iterative in nature and begin with a set of initial solutions. In each iteration a sequence of selection, combination and mutation operations are performed on the set of solutions. The final solution is obtained at the end of all the iterations.

The following example is used to illustrate the working of genetic algorithm.

A chromosome is used to represent a solution to a problem. In this example, a string of bits of a predefined length is considered as a solution. The following steps are performed in a genetic algorithm:

1. An initial population of N random chromosomes is created. This represents the N initial solutions.
2. The initial population is scored based on how good each solution is in solving the problem.
3. Two members of the population are selected for crossover, using the Roulette wheel selection. The higher the score of a solution, the greater is its probability of being selected for the combination. The Roulette wheel method is explained in detail in section 4.3.2.
4. The crossover is performed as follows. Consider two solutions S_1 and S_2, represented using strings of 10 bits as follows:
   S_1: 0110010011
   S_2: 1010011010

First a bit position is determined, called the crossover point for both the solutions. The first child (C_1) is obtained by copying the bits from the first parent (S_1) starting from the first position up to the crossover point. After the crossover point, the bits are copied from the other parent (S_2) starting from the bit after the crossover point up to the last bit. Similarly, the second child (C_2) is obtained by copying the bits from S_2 starting from the first bit position till the crossover point. The remaining bits are copied from S_1.

In this example, assume that the crossover point is 4 for both solutions.C_1 is produced by copyingS_1 till bit position 4. The remaining bits of C_1 (from bit 5 to bit 10) are copied from the corresponding bits (bit 5 to bit 10) of S_2. Similarly, C_2 is produced by copying bits 1 to 4 of S_2 and bits 5 to 10 of S_1. This process is illustrated below in Figure 12.

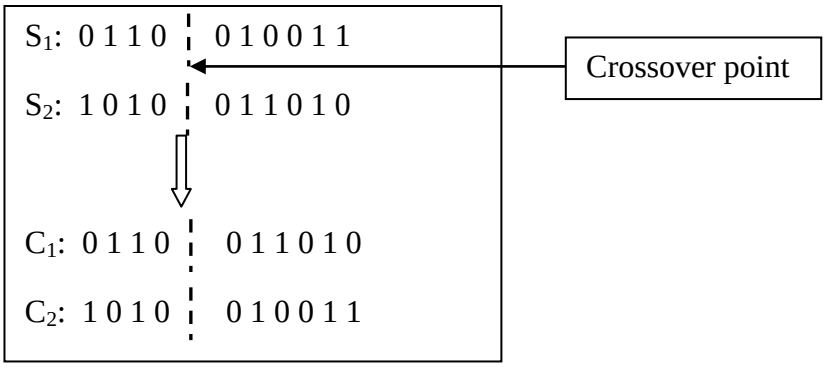


Figure 12: Example of crossover operation

5. The newly created chromosomes C_1 and C_2 are then modified slightly with a very small probability. This process is called mutation. With mutation, the child solution gets a property that is not present in either of its parents. In this example, mutation can be performed by flipping any of the bits.
6. The steps 3, 4 and 5 are repeated till there is no improvement in the solution scores.

4.3.2 Roulette wheel method

This is a method of selecting solutions from the current pool of solutions based on their scores. In genetic algorithms, two solutions are picked from the current pool of solution using Roulette wheel method, in every iteration. Imagine representing the scores of all the N solutions in a pie chart or a Roulette wheel. The angle covered by each solution around the pie is proportional to its score. The higher the solution score, the bigger the slice of pie it occupies. Now the solutions are selected by spinning a ball around the Roulette wheel and grabbing the solution at the point where the ball stops.

This process does not guarantee the solution with the highest score to go through to the next step but it gives all the solutions a fair chance to take part and a good probability that the solution with higher fitness score to move ahead.

5. Analyzing the Z340 using a genetic algorithm

This section presents genetic algorithm to decipher the Z340 cipher, assuming it is a homophonic cipher. Section 5.1 describes how the frequencies of the English letters are created. Section 5.2 describes how the frequency statistics of the symbols in the cipher are created. Section 5.3 details the technique for creating the initial solutions for the Z340 cipher. Section 5.4 explains how the scoring of solutions is carried out. Section 5.5 explains the Roulette method for choosing solutions to perform crossover. Section 5.6 describes the two flavors of crossover: simple crossover and intelligent crossover in detail. Section 5.7 explains the mutation operation. Section 5.8 presents experimental results.

5.1 Creating frequency statistics of English alphabets

In this project, we assume that the Z340 is a homophonic cipher, where each English letter is mapped to multiple cipher symbols (but each symbol is mapped to a single English letter). We use the English letter frequency statistics to create initial solutions for the genetic algorithm. Calculated over large texts, the frequencies of different English letters are different, with some letters occurring a lot more than the others. We use this information, along with the frequencies of the different symbols in the cipher to do an initial mapping and thereby create initial solutions.

For example, if we know that the frequency of the letter ‘a’ in a large English text is about 12%. Then suppose we find 3 symbols Λ, Φ and Ω with frequencies 3%, 8 % and 1 % respectively. Since the sum of their frequencies is equal to 12%, we can initially map the letter ‘a’ to these symbols. In our algorithm, we read the English novel Moby Dick to measure the English letter frequencies.

5.2 Frequency statistics for symbols in the Z340

For the purpose of implementation, each unique symbol in the cipher is assigned to a unique number. This assignment is shown in section 3, Table 1. The frequency of each symbol is calculated in a manner similar to what was done for the English alphabets. The cipher is read and the frequency of each unique symbol is calculated. The number of occurrences of each of the 63 unique symbols in the Z340 cipher is shown in Figure 7.

5.3 Creating initial solutions

Genetic algorithms start from a set of initial solutions. Combination (crossover) and mutation functions are performed on these initial solutions to derive better solutions to the problem. A solution is defined as a mapping from each English letter to a set of cipher symbols. It can also be viewed as a mapping from each cipher symbol to an English letter. The mapping is done in a way such that

1. No Symbol is mapped to more than a single English letter.

2. The frequency of each English letter (calculated as explained in Section 5.1) is approximately equal to the sum of the frequencies of the symbols (calculated as explained in Section 5.2) that it is mapped to.

3. Each symbol in the cipher is mapped to an English letter (no symbol is left unmapped).

To begin, a set of 5 initial solutions is created based on different arrangements of the English letters. For generating a solution, each English letter is considered in some pre-defined order and a mapping is created for it. This mapping is created by matching the frequency of the English letter to the frequencies of the symbols. In our algorithm, we use 5 initial solutions by considering 5 different orders of the English letters. The 5 different orderings of the letters are:

1) Regular order A to Z.

2) Reverse order Z to A.

3) Ordering by increasing letter frequencies.

4) Ordering by decreasing letter frequencies.

5) Random order.

The different orderings of the letters are used to ensure that each letter gets a chance to be mapped to a symbol in the cipher. Our experimental results on the Z408 cipher demonstrate that considering different orderings is better than considering only random orderings of the English letters. The results are better in terms of the number of letters in the putative decrypt matching with the known decrypt.

```
Create_initial_solutions( ) {
    {symbols} = set of all cipher symbols;
    For each english alphabet alpha {
        F_alpha = frequency(alpha);
        Mapping {M} = select symbols from{symbols} such that
                        ( for each symbol S in {M} having frequency S_freq;
                          F_alpha ≈ ∑S_freq;
                        )
        {symbols} = {symbols} – {M};
}
```

Figure 13: Algorithm to create initial solutions

Figure 13 gives the pseudo code to show the steps towards creating initial solutions for a given ordering of English alphabets. In the function Create_initial_solutions the cipher symbols are mapped to English letters. The mapping is done such that the frequency of the English letter is

approximately equal to the sum of the frequencies of the selected symbols. This mapping process is done as follows. A set {symbols} is used to keep track of the set of available symbols for mapping. Initially it is equal to the set of all symbols in some random order. Each English letter is examined and a set of cipher symbols {M} is chosen such that the frequency of the English letter is approximately equal to the sum of the frequencies of the symbols in {M}. The set of symbols in {M} are then removed from the set of available symbols in {symbols}.

Once the process is completed all the unmapped symbols are mapped on to the English alphabets by means of random selection. This ensures that each symbol in the cipher is mapped to an English letter.

Consider the letter 'd' whose average frequency as calculated in [15] is 4.619. There are 3 symbols in the Z340 cipher {8, 29, 43} whose frequencies (calculated as explained in Section 5.2) are 0.882, 1.765 and 1.765 respectively. The sum of these frequencies is 4.412, which is close to the frequency of letter 'd' which is 4.619. In our algorithm, two frequencies are considered close, if they are within 10% of each other. In this example, the set of symbols {8, 29, 43} can be mapped to letter 'd'.

5.4 Scoring putative solutions

Solutions are scored based on the similarity of the putative decrypt to regular English text. There are many ways in which we can measure this similarity. We consider two methods: graph score and word score. Graph scores are a simple and effective way to evaluate solutions without looking up a dictionary. To compute graph scores substrings of different lengths are considered and their frequency of occurrence in a large English text is calculated. This is explained in detail in Section 5.4.1. Word scores are related to the number of English words that can be found in the putative decrypt. English words are identified from the putative decrypt using dictionary lookup. This is explained in detail in Section 5.4.2.

5.4.1 Graph scores

Graph scores are a combination of digraph scores (2 consecutive letters), trigraph scores (3 consecutive letters) and quadgraph scores (4 consecutive letters). Graphs corresponding to bigger substrings (pentagraphs, hexagraphs, etc) can also be considered, but in our experiments, they did not make much of a difference. Each graph score is computed separately and are then added up to create the final graph score. To compute graph scores, a large English text such as a novel is read and the number of occurrences of all distinct substrings (of lengths 2, 3 and 4) is computed. This data is stored in digraph, trigraph and quadgraph respectively. A solution is scored by first generating the putative decrypt, and then generating all substrings of lengths 2, 3 and 4. The score of each substring is computed from the corresponding graph (digraph, trigraph or quadgraph). The sum of scores of all substrings of length 2 becomes the digraph score. The

sum of the scores of all substrings of length 3 becomes the trigraph score and the sum of the scores of all substrings of length 4 becomes the quadgraph score. After computing the digraph, trigraph and quadgraph scores, the graph score of the solution is calculated as a weighted sum of these scores using Equation 2.

$$Graph_score = 2 * digraph_score + 3 * trigraph_score + 4 * quadgraph_score$$

Equation 2: Equation to calculate the solution score

The weighted sum is used to give more weight to the substrings of greater length. The weight given is directly proportional to the length of the substring.

5.4.2 Scoring using dictionary lookup

Another method to score a solution is based on the number of English words that are found in the putative decrypt. Substrings in the putative decrypt are considered and a dictionary lookup is performed to check if the substring forms a word that is present in the dictionary. The score given to a word found is equal to the length of the word. To avoid giving scores to very small words (<= 2 letters), scoring is considered for words of three or more letters. The Ternary search tree data structure is used to quickly search for words in the dictionary. More details about the search mechanism are given in detail under Section 5.6.2.1.

5.5. Crossover using Roulette method

In each iteration of the genetic algorithm, two solutions are picked up from the pool of current solutions and are combined (crossover) to produce two children that replace those two selected solutions. The selection process is done using the Roulette wheel explained in Section 4 under the genetic algorithms. The score of the solution plays a key role in the selection process. The higher the score of a solution, the higher the probability it is selected for the crossover. The area on the Roulette wheel for a solution is proportional to its score. Consider the following example.

Example: Solution 1 has a score 250

Solution 2 has a score 200

Solution 3 has a score 24

Solution 4 has a score 300

Solution 5 has a score 45

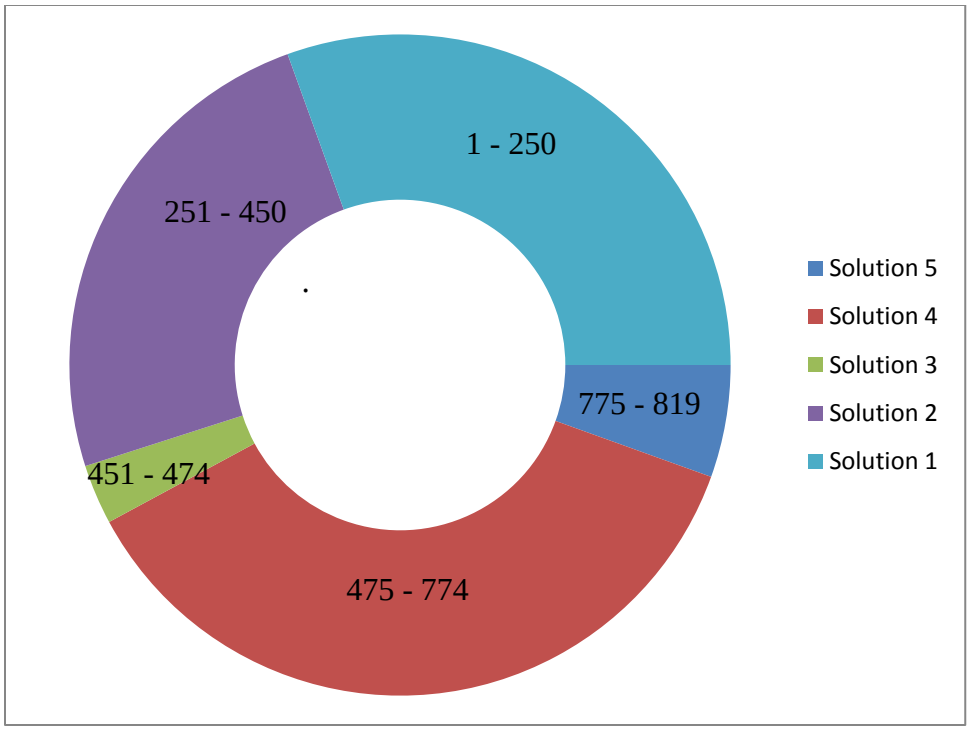


Figure 14: Pie-chart representation of solutions

In Figure 14, there are 5 sections on the pie-chart corresponding to the 5 solutions. The area of each section is directly proportional to the score of the solution it corresponds to. Solution 1 occupies 31% of the pie-chart, Solution 2 occupies 24%, Solution 3 occupies 3%, Solution 4 occupies 37% and Solution 6 occupies 5% of the total area of the pie-chart. In the Roulette method a ball is rolled around the wheel and the place where the ball stops is noted. The solution corresponding to the section of the wheel where the ball stops is the one that is selected.

In our project, the Roulette wheel selection is implemented by assigning score ranges for each solution. This score range is obtained from the accumulated scores. For the example considered in this Section, the score ranges for the solutions are shown in the pie chart in Figure 14. The accumulated score for all the solutions is 819. A random number between 1 and 819 is generated and the score range that it belongs to is noted. The solution corresponding to this score range is the one that is selected. For example, assume that the random number generated is 500. This falls within the range (475-774), which corresponds to Solution 4. After picking the first solution, the second solution is chosen in a similar manner. The second solution is selected such that it is different from the first solution.

The Roulette wheel gives a fair chance for all the solutions to take part in the selection process. The solution with a higher score has a higher probability of being selected for the cross over.

5.6 Crossover

In crossover two parent solutions are used to generate two child solutions. The two solutions that have been selected using the Roulette method are the parents, P1 and P2. The crossover produces two children, C1 and C2 which are derived from parents, P1 and P2. Initially, C1 and C2 are both clones of P1 and P2 respectively. C1 and C2 are modified step by step during the crossover operation. There are two variants of crossover that we have implemented: simple crossover and intelligent crossover. These are explained in Section 5.6.1 and Section 5.6.2 respectively.

5.6.1 Simple crossover

In simple crossover two new child solutions are produced from two parent solutions. In simple crossover, the score that is considered for accepting moves is only the graph score that is explained in Section 5.4.1.The child solutions produced have scores equal to or higher than the parent solutions. Initially the children C1 and C2 are clones of their parents P1 and P2 respectively. Each symbol to English letter mapping in solution C1 is considered and is changed to the corresponding symbol to English letter mapping in solution P2. If this move improves the graph score of C1, then this move is accepted. If this move fails to improve the graph score, then the symbol's mapping is restored to the original mapping. This process is continued for each symbol in the cipher. The same procedure is done for C2, this time changing the symbol to English letter mapping with P1.

After the two new children C1 and C2 are obtained with scores higher or equal to their parent scores, the parents P1 and P2 are replaced with the children C1 and C2.

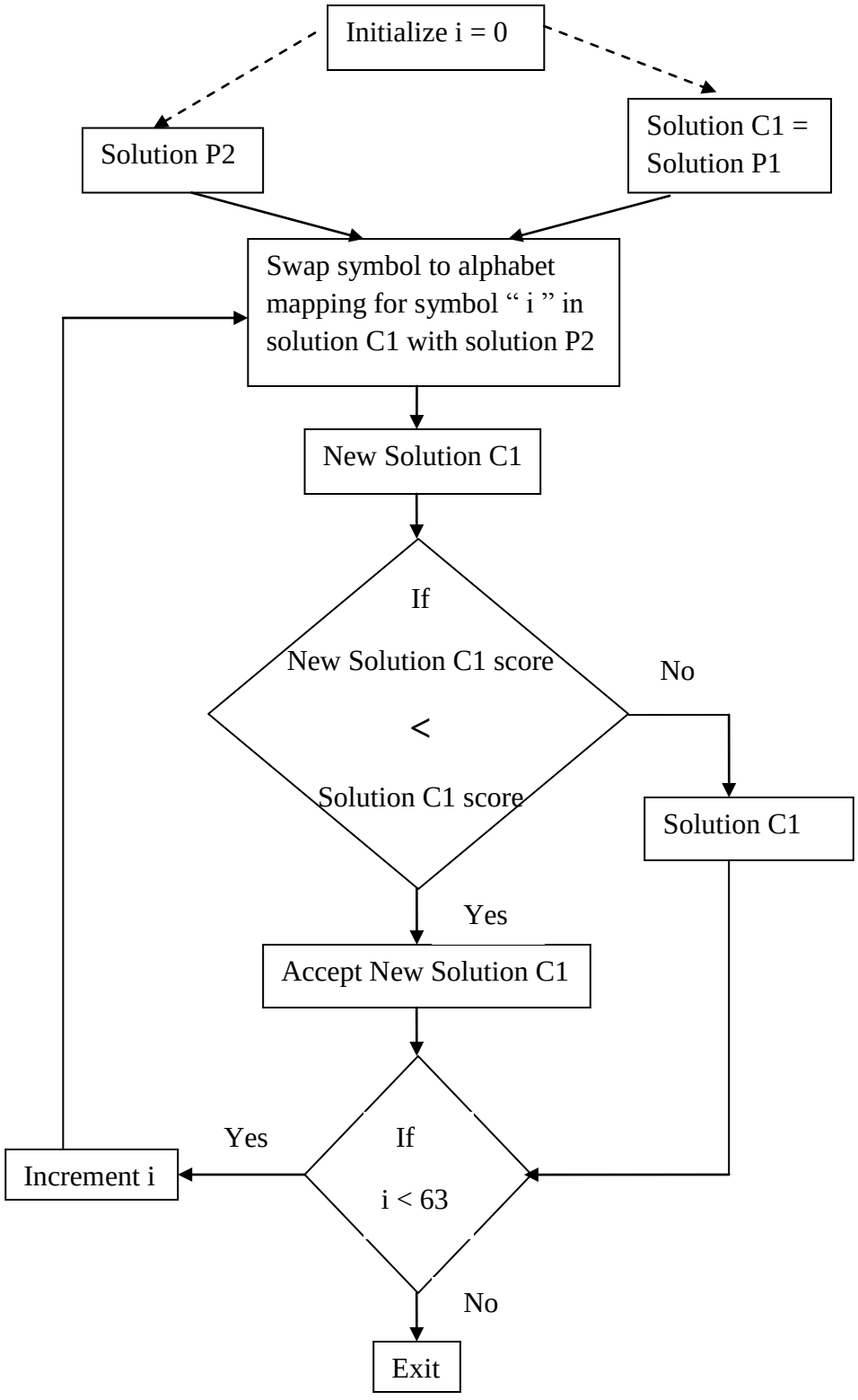


Figure 15: Flow chart for simple crossover (producing child C1 from parent P1 and P2)

5.6.2 Intelligent crossover

The previous Section 5.6.1 presented a simple crossover technique using graph scores. In order to generate solutions with more English words, it is necessary to evaluate how many English words can be found in a decrypt generated by a solution. The intelligent crossover technique uses this information, in addition to the graph score for evaluating solutions. Furthermore, symbol to English letter mappings that yield English words of length three or more are locked from being changed during crossover. The flowchart for intelligent crossover is shown in Figure 20. The main differences between the simple crossover and the intelligent crossover are as follows:

1. The scores that are considered when making decisions are not just graph scores, but a combination of graph scores and word scores.

2. Symbols that combine to form a word in the dictionary (of greater than or equal to three letters) are locked from being swapped in future iterations. This is to preserve symbol to English letter mappings that form English words.

To explain the working of the intelligent crossover method, the Ternary search tree data structure for storing the English words (dictionary) and the algorithms for searching for words is first explained (Section 5.6.2.1). The actual technique for intelligent crossover is explained in detail in Section 5.6.2.2.

5.6.2.1 Ternary search tree (TST)

A dictionary of English alphabets is built for the intelligent crossover. The Ternary search tree data structure is used to store and search English words [18]. The Ternary Search Tree (TST) is used as it occupies less space and makes it faster to identify/ find words [19].

The TST has a Ternary tree data structure. It has a left child, equal child and a right child. Each node stores a single English letter (called splitchar), a data field and a reference (pointer) to each of the three children [10]. The data field of a node is null unless the sequence of splitchars starting from the root node until the current node forms a word in the dictionary. The word formed becomes the data stored in the node.

When traversing the TST to search for a word, the traversal is made depending on whether the current character being matched is less than, equal to, or greater than the splitchar in the current node. If the character is less than the current node's splitchar, the left child becomes the next node. If the character is equal to the current node's splitchar, the equal to child becomes the next node. If the character is greater than the current node's splitchar, the right child becomes the next node.

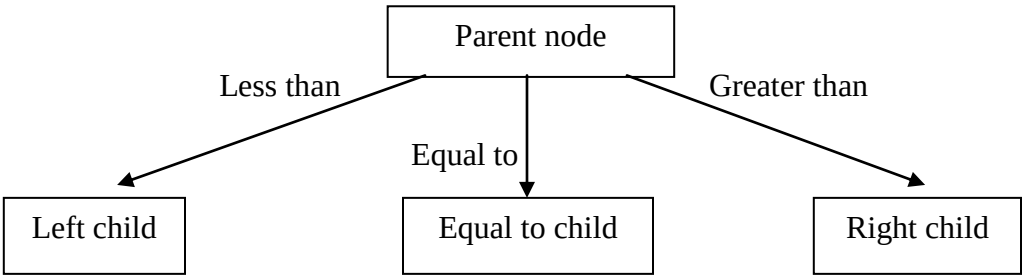


Figure 16: Ternary tree

Inserting words to the dictionary

```
Insert_word(word, index, current_node) {
  if(current_node.splitchar == NULL) {
     current_node.splitchar = word[index];
  }

  if(current_node.splitchar == word[index] ) {
     if(index == word.length() -1 ) {
        current_node.data = word;
        return;
     }
     if(current_node.equal_kid == NULL) {
        current_node.equal_kid = new Node();
     }
     current_node = current_node.equal_kid;
     index++;
  }

 else if (word[index] > current_node.splitchar ) {
    if(current_node.high_kid == NULL) {
       current_node.high_kid = new Node();
    }
   current_node = current_node.high_kid;
 }

 else {
    if (current_node.low_kid == NULL) {
      current_node.low_kid = new Node();
    }
     current_node = current_node.low_kid;
 }

 Insert_word(word, index, current_node);
}
```
Figure 17: Algorithm to insert word in the dictionary

Figure 17 illustrates the algorithm to insert a word in the dictionary. The function Insert_word is a recursive function that is called with the following arguments: (i) the word to be inserted, (ii) the index of the current letter in the word being inserted and (iii) the current node in the Ternary tree. The splitchar represents the character stored at the node. If the current node's splitchar is not assigned, then it is assigned to the English letter word[index] (the character at the indexth position in word). If all the letters of the word have been processed, then the data field of the current node is set to the word and the function is returned. This marks the end of insertion of the word in the Ternary tree. If all the letters of the word are not assigned, then the current node is assigned to the equal kid of the current node and the index is incremented by 1 and the Insert_word function is called again.

For the case where the English letter word[index] is greater than the current node's splitchar, the current node is set to the present current node's high kid (right child) and the Insert_word function is called again. The remaining case is when the English letter word[index] is less than the current node's splitchar. In this case the current node is set to the present current node's low kid (left child) and the Insert_word function is called again.

An illustration of how the dictionary is build is shown below. Consider the words "kill", "kit" and "key" to be inserted in the dictionary.

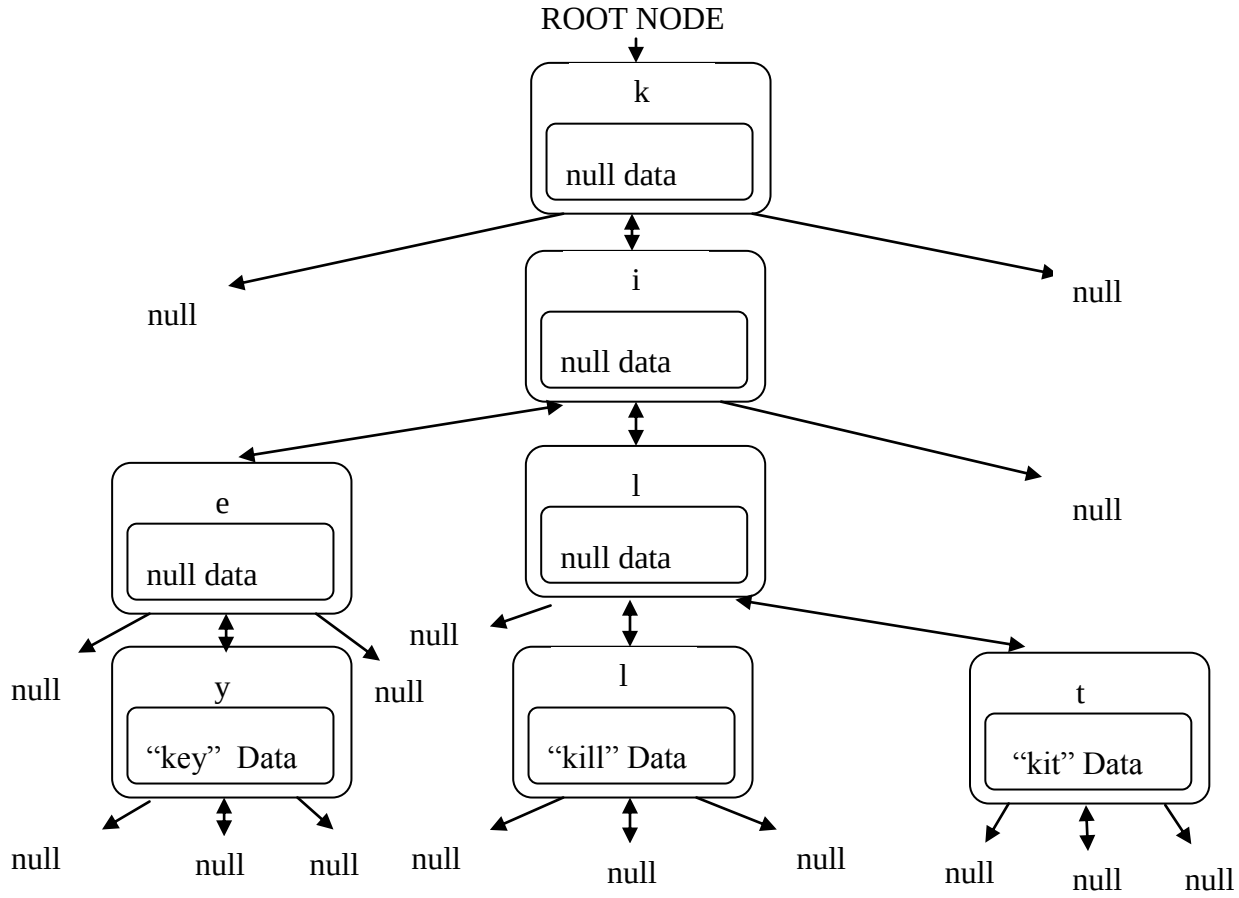


Figure 18: Representation of words in the dictionary

The words are inserted one at a time. When inserting a word, the root node corresponding to the first letter of the word is located. If the node cannot be located, it is created. The function Insert_word is called with the word to be inserted and this root node. In this example, the word “kill” is first created. The function insert_word is first called with word “kill”, index 0 and the root node corresponding to the letter ‘k’. Since this is the first time a word is being inserted that starts with ‘k’ a new node is created and attached to the root node, the node’s splitchar is unassigned. Therefore the splitchar is then assigned to ‘k’. This node’s equal kid is assigned to NULL, so a new node corresponding to the equal kid is created. The index into the word is incremented and the Insert_word function is now called with the new node that was created. This node’s splitchar is also unassigned, and it is now assigned to the next letter in the word ‘i’. The same procedure is continued till the last letter of the word ‘l’ is encountered. Here since ‘l’ is the last letter, the data field of the node is assigned to the word “kill”, and the function is returned. This completes the insertion of the first letter “kill” in the dictionary.

Now consider the insertion of the second word in the dictionary “kit”. The root node corresponding to “kit” is for the letter ‘k’ which has already been created when the first word “kill” was inserted. The function Insert_word is called with the word “kit”, index 0 and this root node. The current node’s splitchar is the same as word[index], and its equal kid has been created during insertion of the word “kill”. The function Insert_word is called with “kit”, the index 1 and with the current node assigned to the equal kid of the root node. Again, the current node’s splitchar is equal to the word[index], and the Insert_word function is called with the word “kit”, the index 2 and the current node assigned to the previous current nodes’ equal kid. Now, the current node’s splitchar (letter ‘l’) is not equal to word[2] (letter ‘t’). In this case ‘t’ is greater than ‘l’, so the letter ‘t’ has to be inserted along the path of the current node’s high kid. Since the current node’s high kid is assigned to NULL, a new node is created and is attached to the current node. The function Insert_word is now called with the word “kit”, index 2 and the current node assigned to the previous current node’s high kid. Now, this node’s splitchar is not assigned, and therefore the splitchar is assigned to word[2] (letter ‘t’). This being the last letter in the word, the data field of the node is assigned to “kit” and the function is returned.

Now consider the insertion of the word “Key”. After processing letter ‘k’, consider the stage when the Insert_word function is called with the word “key”, the index 1 and the current node set to the equal kid of root node ‘k’. The word[1] (letter ‘e’) is less than the current node’s splitchar (letter ‘i’). A new node for the current node’s low kid is created and the Insert_word function is called with the word “key”, the index 1 and the current node assigned to the previous current node’s low kid. Now the current node’s splitchar is not assigned, and is therefore assigned to word[1] (letter ‘e’). A node for the equal kid is created and Insert_word is called again with word “key”, index 2 and current node assigned to the previous current node’s equal kid. Finally the letter ‘y’ is assigned to the current node and the data field is set to “key”. This completes the insertion of the word “key” in the dictionary.

Finding words in the dictionary

The Ternary tree data structure makes it very fast to check if a particular word exists in the dictionary or not. It can be quickly determined if there is a word starting with a given set of characters (prefix search).

```
int Find_word (word, current_node, index) {
    if (current_node == NULL) {
        return 0;
    }

    if(index == word.length - 1) {
        if(word[index] == current_node.splitchar) {
            if(current_node.data != NULL) {
                if(current_node.data == word) {
                    return 1; // word found
                } else {
                    return 0; // word not found
                }
            } else {
                return 2; // word can be found
            }
        }
    }
    if(word[index] < current_node.splitchar) {
        x = Find_word(word, current_node.low_kid, index);
    } else if (word[index] > current_node.splitchar) {
        x = Find_word(word, current_node.high_kid, index);
    } else {
        index = index + 1;
        x = Find_word(word, current_node.equal_kid, index);
    }
    return x;
}
```

Figure 19: Algorithm to find word in the dictionary

Figure 19, illustrates the algorithm to find if the given letter sequence forms a word in the dictionary. The function Find_word is a recursive function that is called with the following arguments: (i) the word (sequence of letters) to be found in the dictionary, (ii) the index of the current letter in the word being found and (iii) the current node in the Ternary tree. If the current node is not assigned, then there is no word with this letter sequence in the dictionary. The splitchar represents the character stored at the node. If all the letters of the word have been processed, the current node's splitchar is compared with word[index] (indexth letter of word) and if there is a match found, check is performed to see if the current node's data field is empty. If the data field has null stored, it implies that the given sequence of letters is not a word but is a

prefix to another word or words and a flag is returned to indicate that a "word can be found". Prefix search is useful during the process of searching for words in a putative decrypt. Once we know that there is a word that can be found, we can keep searching for words with that prefix. Furthermore if we find out that a word cannot be found we can stop looking for words with that prefix.

If the data field is not null, a check is performed to see if the given word (sequence of letters) is stored in the current node's data field. If this is the case, we know that a word is found and we return a flag to indicate that a "word is found". If the current node's data field does not match the given word string then, we know there is no word in the dictionary with this order of letters and we return "word not found". This marks the end of finding a given word in the Ternary tree.

For the case where the character word[index] is less than the current node's splitchar, the current node is set to the present current node's low kid (left child) and the Find_word function is called again. For the case where the character word[index] is greater than the current node's splitchar, the current node is set to the present current node's high kid (right child) and the Find_word function is called again. The remaining case is when the character word[index] is equal to the current node's splitchar. In this case the current node is assigned to the equal kid of the present current node and the index is incremented by 1 and the Find_word function is called again.

An example of how the Ternary search dictionary tree is searched to find the word "key" is illustrated below.

Consider the Ternary tree that has been built in Figure 18.

When finding a word, the root node corresponding to the first letter of the word is located. The function Find_word is called with the word to be inserted and this root node. In this example, the word "key" is being searched in the dictionary. The function Find_word is first called with word "key", index 0 and the root node corresponding to the letter 'k'. Here the word[index] (letter 'k') is equal to the current node's splitchar (letter 'k'). The index is incremented by 1 and the Find_word function is called with the current node set to the current node's equal kid.

Now the word[index] (letter 'e') is less than the current node's splitchar (letter'i') and the Find_word function is called again with the current node set to the current node's low kid. Now the word[index] (letter 'e') is equal to the current node's splitchar (letter 'e'). The index is incremented by 1 and the Find_word function is called with the current node set to the current node's equal kid.

Now the last letter of the input string has been reached and the word[index] (letter 'y') is equal to the current node's splitchar (letter 'y') and the current node's data field is equal to the input word string that is being searched. Therefore the word has been found and the search is complete.

5.6.2.2 Intelligent crossover technique

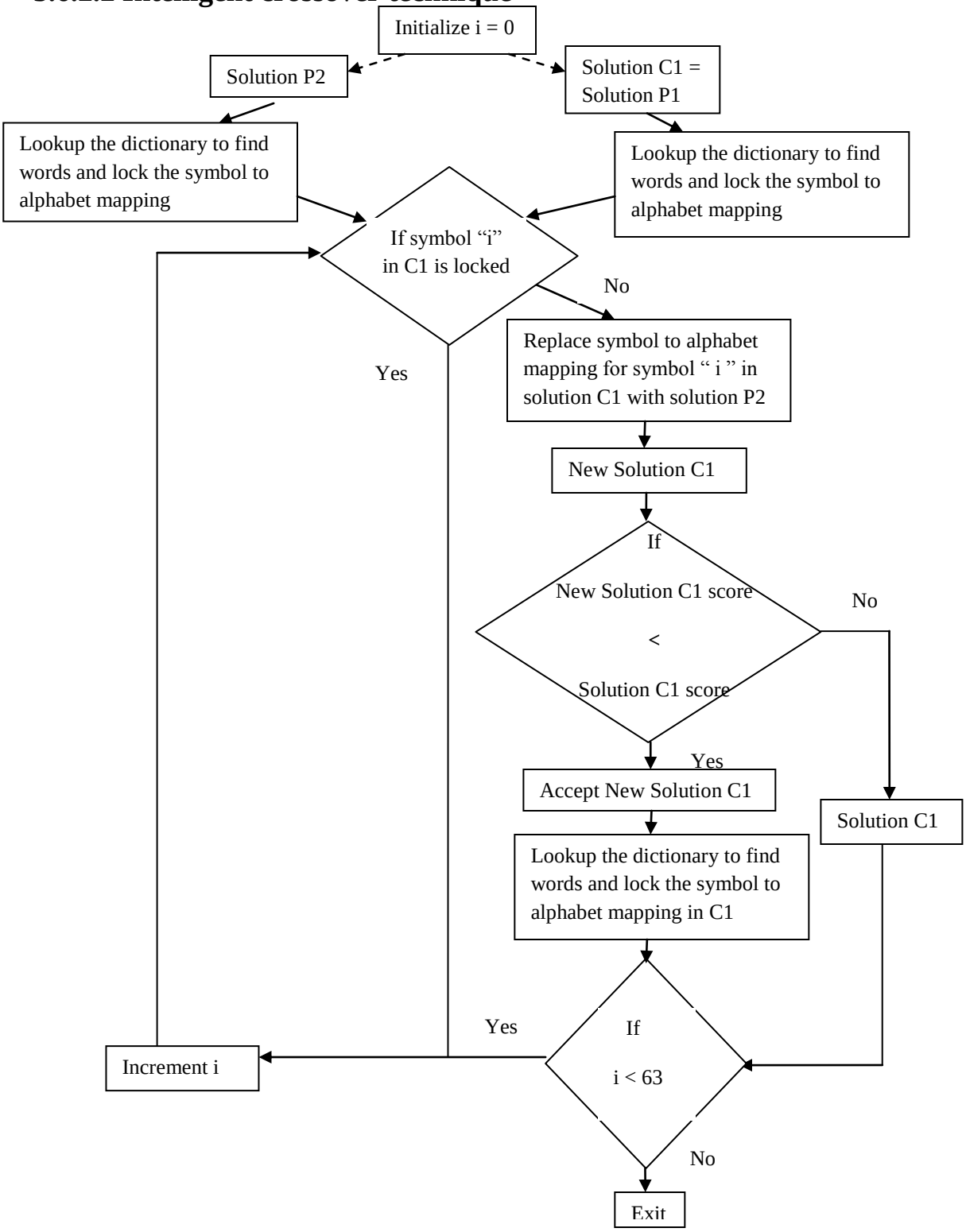


Figure 20: Flow chart for intelligent crossover (producing child C1 from parent P1 and P2)

The flow chart for the working of the intelligent crossover is shown above in Figure 20. In intelligent crossover two new child solutions are produced from two parent solutions. In intelligent crossover, a normalized score is considered for accepting moves. The Normalization process for the graph score and the word score is explained in Section 5.6.2.3.The child solutions produced have scores equal to or higher than the parent solutions. Solution P1 and solution P2 are picked using the Roulette method for crossover. The crossover produces two child solutions C1 and C2. Initially C1 and C2 are the clones of solutions P1 and P2 respectively.

The dictionary lookup is performed on both solution P2 and solution C1. The dictionary lookup reads strings of size 3 or more characters from the putative decrypt text. The Find_word function explained under section 5.6.2.1 is used to search for words. Each time a word is found, for each letter in the word, the corresponding symbol to English letter mapping is locked.

The crossover is performed one symbol at a time. For each symbol, a check is made to see if the corresponding symbol to English letter mapping in solution C2 has been locked. If it has been locked then the next symbol is examined. If the symbol to English letter mapping is not locked, then the symbol to English letter mapping for that symbol in solution C1 is replaced with the corresponding symbol to English letter mapping from solution P2. A decrypt for the cipher using this modified solution C1 is obtained. This is scored using the scoring algorithm explained in Section 5.6.2.3. If the score improved after the change in the mapping, the move is accepted, otherwise the original mapping for the symbol is retained. If the modified solution C1 is accepted, dictionary lookup is performed on the decrypt obtained from this solution and the symbol to English letter mapping for the letters in all the identified words are locked. This procedure is repeated for each symbol in the cipher. Similarly solution P1 and child solution C2 are considered for crossover process. After the two new children C1 and C2 are obtained with scores higher or equal to their parent scores, the parents P1 and P2 are replaced with the children C1 and C2.

The Roulette method is again called and 2 solutions are picked from the list of solutions for crossover. The same procedure is followed for these 2 solutions. This process is repeated until there is no improvement in the scores.

5.6.2.3 Intelligent word score (Score normalization)

The score that is considered for accepting moves in the intelligent crossover is called the intelligent word score. It is a combination of graph score and word score. Graph score is explained in Section 5.4.1. Word score of a solution is related to the English words that are found in the corresponding decrypt. Each English word found in the decrypt contributes a score equal to the number of letters in the word. The sum of all such scores produces the word score for the solution. For example the word “the” will contribute a score of 3 and the word “hiding” will contribute a score of 6.

$$\textit{Intelligent word score} = \textit{Normalized Word Score} + \textit{Normalized Graph Score}$$

Equation 3: Equation to calculate intelligent word score

Graph scores depend on the number of occurrences of substrings in large texts. Therefore the graph scores can be very large. We read the English novel Moby Dick, to record the substring frequencies. In our experimental results, we observed graph scores to be up to a maximum of 100,000. On the other hand, the word scores are limited to the length of the cipher. For the Z340 cipher, the maximum word score is 340. To give equal weight to the graph and word scores, these scores have to be normalized. The normalization is achieved by considering each score as a percentage of the maximum possible score. Finding maximum possible word score is easy. The maximum possible word score is simply equal to the number of characters in the cipher, which in this case is 340. The maximum score is achieved when each character in the cipher belongs to some English word that can be found in the dictionary. In other words, a word can be found to cover each character in the putative decrypt.

$$\textit{Normalized Word Score} = \frac{\textit{word score for a particular solution}}{340} * 100$$

Equation 4: The formula to calculate the word score

The maximum possible graph score is found through experiments. We run the genetic algorithm using only simple crossover for an unbounded number of iterations, till the solution score show no improvement. We repeated this experiment many times and recorded the maximum graph score over all the runs.

$$\textit{Normalized Graph score} = \frac{\textit{Graph score}}{\textit{Maximum Graph score}} * 100$$

Equation 5: The formula to calculate the graph score

To illustrate the calculation of the intelligent word score, consider the following scenario. Assume that the maximum possible graph score is 100,000. Now for the particular solution, assume that the graph score is 70,000. Also assume that the word score is 170. The normalized word score is calculated from Equation 4 as 170/340*100 = 50. The normalized graph score is calculated from Equation 5 as 70,000/100,000 * 100 = 70. The intelligent word score is therefore 70 + 50 = 120.

5.7 Mutation

In genetic algorithms, crossover is followed by mutation where the child solutions are modified slightly with a very small probability. Thus it is possible that the children have slightly different characteristics than either of the parents. Mutation is performed after both simple crossover and intelligent crossover by modifying a symbol to English letter mapping with a probability. After crossover is performed a symbol is picked randomly among the unlocked symbols. Assume that the probability of mutation is p (where typically p is around 0.001). A random number R is generated between 0 and 1. If this number is less than or equal to p, mutation is performed, otherwise it is not performed. Mutation is performed by randomly assigning an alphabet from any of the 26 English alphabets to the selected symbol.

5.8 Experimental results

5.8.1 Different flows of the genetic algorithm

We ran our experiments using different flows for the genetic algorithm. The flowchart for the basic flow is shown below in Figure 21.

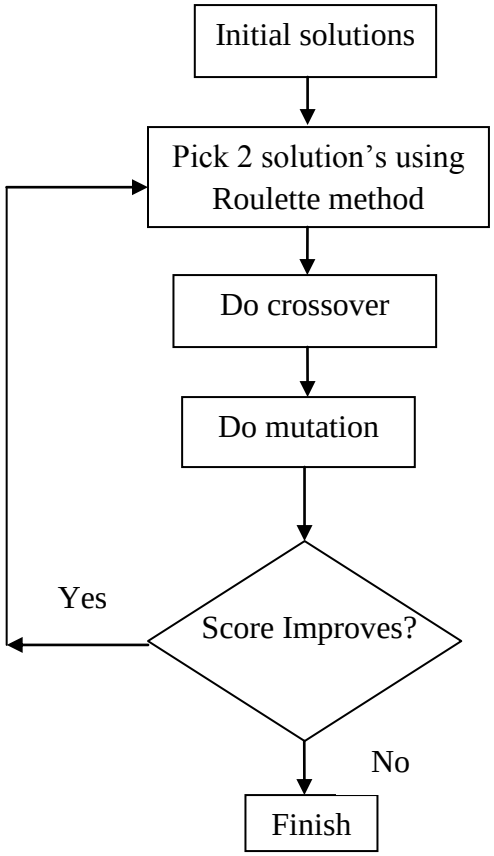


Figure 21: Flowchart for genetic algorithms

The crossover is performed using either simple crossover or intelligent crossover. The different flows that were tried are:

1. Using only simple crossover: In this flow, only the simple crossover is used for the crossover operation. The iterations are done till there is no improvement in the scores of the population.

2. Using only intelligent crossover: In this flow, only the intelligent crossover is used for the crossover operation. Again, the iterations are done till there is no improvement in the scores.

3. Using a combination of simple and intelligent crossover: In this flow, the initial iterations are done with simple crossover. The remaining iterations are done using intelligent crossover, till there is no improvement in the scores.

4. Using cribs: Cribs are putative decrypts of portions (usually small) of the cipher. They act as hints in the decoding of the cipher. We performed experiments using the above 3 flows with and without the use of cribs. For the Z408 cipher, which has a known decrypt, we gave a couple of words as cribs that we knew to be correct. The Z340 cipher does not have a known decrypt. Therefore we guessed a couple of words and provided them as cribs to the Z340 cipher.

5.8.2 Testing genetic algorithm on the Z408 cipher

We tested the working of our genetic algorithm on the Z408 cipher which has a known decrypt. We used the different flows as described in section 5.8.1 and measured the number of characters that matched with the known decrypt. We performed a number of runs for each flow and recorded the solution which had the maximum number of matching letters with the Z408 decrypt.

Table 3: Experiments on Z408 cipher

Crossover method	**Crib used**	**# of iterations**	**# matching characters with known Z408 decrypt**
Simple	---	36	52
Intelligent	---	46	68
Simple + Intelligent	---	41	102
Simple	give	55	104
Intelligent	give	50	103
Simple + Intelligent	give	41	123

When cribs were not used, the best solution was obtained with the combination of simple and intelligent crossover. The number of matching characters with the known Z408 decrypt was 102(25%). Using only simple or intelligent crossovers did not achieve such a good result, and gave best matching of 12.7% and 16.7% respectively.

The decrypt for the best solution in terms of number of matching characters with the Z408 cipher obtained using both simple and intelligent crossover is shown below (English words in this decrypt are printed in capital letters):

elnrerinliSEEeleneeeaehaietlvltehahhhtiliREALecheTARerelnnse
iilaeheeitohehTAREATeEARhveeeeellaiELITEreeellhehSEEenacrlNO
TrnlnvleetHITeeineleelaeeTITohaelnneeesEIREeaeitiaeneSEEltel
oahteitleeeeaHARTarvlhcinthieILLoaeeelteeALTHITeiohheiaeenei
eiielneereealsntERAhaiAIREeelllheiarnerelleainlneealeeeeALIN
EveienlSATeineeTHEETREEeeaehleeahielloLETTANleeLIEtrroteeEAT
lliatnselcalenevHALEeihLEANnceieelaeeLIEethaeili

We also performed experiments with the use of cribs. We used the word “give”, which is encoded using 4 symbols in the Z408 cipher. These symbols cover 38 characters in the cipher text. With the use of the crib, the scores improved for all three crossover methods. The scores for both the simple and only intelligent crossovers increased to around 25.5%. The score for the method using simple and intelligent crossover, increased slightly to 30.1%. The decrypt for this solution is shown below (English words in this decrypt are printed in capital letters):

tliteTIElingtnlterBEEeterteleeeetehetnieieeinlathleaittleing
liLAGreLINTheeerineebreeteeeehteeereleleaiglNETlrnteeeiaaLET
eTILEelelehingGIVEeeneereeelthrtleihgestriLIELIEieevnnbRELEN
TernGREEThgTITrieetelealilheginltelbleetereeeiltethrlleeiiei
niltlebrieBINninTARraiereHEELlehLIEavlttllealilebneleretelEV
EetltelnilGIVEtetetnaerBEEetelTITltlltnteeeelTELLheietetetee
llrelinglaelEVEeeiNETeeeereiarrbllrtrereleheTIEi

Below, we provide another example decrypt for one of the solutions. A number of interesting words were found in this decrypt. The English words in this decrypt are printed in capital letters. This solution is obtained using only the simple crossover, without the help of any cribs. A total of 104 matches were found with the known Z408 decrypt (25.6%).

eLETeTILLINGoelolSEEtLETHEELelnietelenirioienleeheersTELLeng
eilsGAINineeeLIEeeteesTOEeeilhelreHILLEARSglnrenaneILLeerlle
iTELLeLINEeingGIVELIERESINleeeeellehgecohelstnieitEVENesreen
eeangHEREhgneeeeitTELLeeeeelginleelenleolernLIEeleealeeeSEAi
eieelleSEEeenneNOReaSIThoHALLlreLIErvnTELLeSEElleetliSINtllv
eeeeellneeGIVENieinnRISEetlelnneeeellennentllealehieoenoinTI
LLhTEENgLETllveeLENINllREELeeSHEnleesRHINEeeoiri

Analyzing the decrypt above, we found a number of interesting matches with the known Z408 decrypt. For example, in the first line, the word “TILLING” is very close to the actual word

"KILLING" in the known decrypt. The word "EVEN" at the end of line 3, correctly matches to the known decrypt. There is one instance of the word "GIVE" on line 3 and an instance of the word "GIVEN" on line 6. The word "GIVEN" on line 6 is because of the use of the crib. But the word "GIVE" on line 3 is not entirely because of the crib. The letters 'I' and 'E' are from symbols other than those given for the crib.

5.8.3 Results for the Z340 cipher

We performed a number of experiments using our genetic algorithm on the Z340 cipher. We used all the flows described in section 5.8.1. Since there is no accepted decrypt for the Z340 cipher, we evaluated the genetic algorithm by observing the English words that were detected in the various decrypts of the solutions.

5.8.3.1 Experiments without cribs

We first ran the experiments without providing any cribs. We used 3 different crossover mechanisms, as described in section 5.8.1. We measured the word scores of the decrypts obtained from the different solutions. We did a number of runs and recorded the solutions which gave the best word scores. Word score of a solution is the weighted sum of the number of English words found in the decrypt. The results are detailed in Table 4 below.

Table 4: Experiments on Z340 cipher without using cribs

Crossover method	# of iterations	Word score
Simple	35	195
Intelligent	60	163
Simple + Intelligent	38	169

From the table, we can observe that the simple crossover gave the best results in terms of the word score compared to the other crossover techniques. Even though using simple crossover produced the best score, we observed that most of the words were 3 or 4 letters long, and many words appeared multiple times. The decrypt for the best solution using only simple crossover is provided below.

hteeeaSATennaANTIeelteHARTnelnTHElthrenlrsnnelaheHEREeiSANTA
LEALLehTHENLENTiNETTLEeeTENestaaeTILEtettehelharotreinaLEANe
nTEETHalheeeleaeinnlalSETtteHOVEheEARalselaeintTHErLETentnte
TEAanHATneaeenTEAenlLETtEVERhhleNETteeONEANTESEETEAleaenllhT
INTREELTEARSeelnelLETheaeslenANTEealaetlnlTANtnlALLlnleNETar
HATELENTeinNINTHEATteleenTHEEereSANonhol

It can be observed from the decrypt that most of the words start with T, S or N. Furthermore the decrypt is dominated by a few letters ‘t’, ‘e’, ‘a’, ‘l’, ‘n’ and ‘s’. Such a distribution is unusual for a meaningful English text.

Next we will consider a decrypt obtained from the best solution using a combination of simple and intelligent crossovers.

ssLETTAILHATEthaeitkSLAINeeeenlALESHATEkleheleEARSeettSATeLI
PtikklaaheekREVEALilkorhetelaiTITLEahieaelalkatntALTihieREAL
haelvaikhierkteeenekokelaisthltRATrtlikaeKITSnishtlklelesaee
stoTEAiANTtleALLtLEEKoSIRteehakiaLISTlthiiALLALOETILEkkhS
ITlAREASeileltkATEprsalitaeLETHALttkelaanKITESaeekkeEARetLIE
steLETaiheenseSHEislTATeeeSITHEREielaatk

From the sample decrypt, we can observe wide range of words of different lengths, such as “SLAIN”, “HATE”, “RAT”, “REVEAL”, “LETHAL”, “KITES” etc. Furthermore, we see a more realistic distribution of letters in the decrypt.

5.8.3.2 Experiments using cribs

We performed experiments on the Z340 cipher using cribs. From the original cipher (Figure 4, the first three characters looks like “HER”. Also the characters 10 to 15 of the last line look like “ZODAIK”, which could actually be “ZODIAC”. We used the words “HER” and “ZODIAC” as cribs. The cribs were represented using 9 symbols and they covered 52 characters in the cipher. We ran our genetic algorithms using these cribs and recorded the best solutions, in terms of the word scores. These scores are tabulated below:

Table 5: Experiments on Z340 cipher using crib (HER, ZODIAC)

Crossover method	**# of iterations**	**Word score**
Simple	35	136
Intelligent	77	159
Simple + Intelligent	45	162

The decrypt for the best solution using simple crossover is shown below:

HERhreielaaaLEDentrlileotthnTENceaitTEAlszdhlaleeHENrrTIEDNE
ARollrceiaalehrnatlllTHANrdrileoinntalhetLETlCITEeseldetelal
deeereolitneltLANEalilzreLIETIThcieeseLIElortelitislrnTHEatn
irieacoeeeereANTETATlTILETANtelTATLIELEDREDlTIEteTITeethllti

lANTHATEhesztilaetaeicloRITEAIDEReellteTELLehiaALLlahtehelon
HENlaralanhethiinoiniteHAThtRANeZODIACel

The decrypt for the best solution using only intelligent crossover is shown below:

HERteiabhaaOILiisSEEqopoCOSTdilcahqoolNETzisoHIPwhTHEEcAIDlB
EEoeerciTONEwenSATnhemlaiedRANioslsaANTIooptectCHITlsibdwiao
iiTANPOETStwerioSINESezrinqlOILlcsWITBEATeoecinqosteRITEeaot
qeslncoiilirtALTltndemqnwloHOPESatnqlohdriinsaaTITSdlbteeeoq
SOLOloaetbtztSEALDEWqcooeadANTIirlleitiaienisqahieehSAWSlhoh
hlioheanASSICEqttoqlSALTnohSEAhwZODIAChe

The decrypt for the best solution using a combination of simple and intelligent crossover is shown below:

HERildnnHEATehNETaLETeTOEHALElecnnTAUTTEAznaeNETeHELLltndDEN
lloeerceLETeeANTAIRheTHEildrnrDONEtrerieheTIEceeeEATinneeeae
neennTOElaleeMEETltEYEzrerttaishcneDANEneeoltlrTANaeriiaeahl
tlyhtcoeltdreaeiHITeettresELATEaAIRtteedrdnRANnieiYETniaeEAT
iteuHEREINaziNEATeLETceolnenTENerTHEEierlerDATAneeenAREAthol
hhienlAREtALTatlLOTenrTIThHALEleZODIACee

The solution obtained using the combination of simple and intelligent crossover produced a word score of 162. In other words, about 47.6% of the characters in the cipher were covered by English words from the dictionary.

A set of runs were performed on the Z340 cipher with the crib “KILL”. We used this crib, because we felt that it was the word most likely used by Zodiac in his cipher. The crib was used at all possible offsets in the cipher, i.e. a total of 337 trials were run. From the original cipher (Figure 4) the first four symbols were locked to the letter K, I, L, L in that order, in the first trial. For the second trial the symbols starting from location 2 to 5 were locked to “KILL”, and so on.

Table 6: Experiments on Z340 cipher using crib (KILL)

Crossover method	**Trial number**	**Symbols locked**	**Word score**
Simple	188	29, 36, 6, 3	178
Intelligent	191	3, 41, 11, 30	186
Simple + Intelligent	326	1, 18, 5, 10	189

Table 6, shows the results with the best score for the Z340 cipher using a genetic algorithm with the crib KILL. In this table, the trial numbers in column 2 correspond to the trial that produced the solution. The column for “Symbols locked” shows the symbols that were locked to the letters ‘K’, ‘I’, ‘L’ and ‘L’ in that order.

The decrypt for the best solution in terms of word score using only simple crossover is shown below:

tALTOlnhtnTIEhTITioetnlLINENekEARhteriheinTENheLETndooinlaeh
loleelaiiiheeHITtTATenanooalnALLeetnnaTINnlTEASiniiilTHEEetn
TINRILLeiineeneitkheaenliaTIEaraaeeliHENneloikATEeieLOTHatnn
toahhaliKILLntethTHEEntaeRIDEleitTATinnALLtainrTITaeihTHEEet
LIERAINaTHINteeTIELETanLONERhstiliheeTINkeaLETTHEEeHENeeitld
thonHOTANTEkihTINlTEENithntIONDENlaaTANe

The decrypt for the best solution in terms of word score using only intelligent crossover is shown below:

aokNETeennlTITtosvedLESSPEThsILLnilTOILdretteiisaAIMeeteteLE
TesddkLOSElDAMNSlcENDlsnteekeetsklshnenoeescdLOPNORieteSAILe
toINNSsdsvHADVIESildfdekoeliTANSlkatREDeIDSeTIEltkrdktcMOLEh
LEFTllsoiitKILLctclsdlLEANemtsdvlceLIEnekttEVENcocfsiecmddtL
ETLOSEhONEReckdLISTALLeSEESnLOToKITdicoHIDEttlliiddiTHATinsM
ATTEieLENSTITmlshsllkhinLEAVENmaeSEALlnd

The decrypt for the best solution in terms of word score using a combination of simple and intelligent crossover is shown below

KENILLliolAILAREhilerHEALrttdsnSEAREDeNETirtHALEekdlllelLENI
NlaeensELANeeeTHATioeaololenlilasnhaliierhetesFLEETeeRIDElah
REDeTEAELITEellahsnereinEIREETHOSselTIElDEALeSIRESTENoteEARt
rlRANsaeselndANTatnDEARIEHALEeeiaTIREheenlriiLETetrdeiteeeer
eindoaaeiiTITSeaednerSHALLDENfreneaelTEASEiltraaleEATaeteoal
kaohalAILhtSEErlTARNSaeinrKILLleiaetasee

The solution obtained using the combination of simple and intelligent crossover produced a word score of 189. In other words, about 55.6% of the characters in the cipher were covered by English words from the dictionary. Even though the crib KILL produced solutions with high word scores, we were not able to obtain any meaningful decrypts.

We have attached some more interesting results in the Appendix B for the Z340 cipher and in Appendix C for the Z408 cipher.

6. Future work

Future enhancements to the project could be as follows:

Ability to predict words given partial words: In our experiments, we found lot of interesting words that were partially formed. But we failed to detect them in our algorithm because we only look for complete words (with correct spellings) in the dictionary. If we can intelligently find incomplete words and change the mapping such that we complete the word, then we can get much better results. For example if we decode a substring as "KILLEM" which is not an English word, but if we change the mapping of the last symbol so that we get "KILLER", then we can do much better.

Provide user control: In my project, everything is automated, from start to finish. It will be useful if the user can stop the iterations in between, take a look at the results found so far and lock some symbol to English letter mappings.

Recognize correct grammar: My project only looks for words and does not view a sentence as a whole. If we can think about a grammatically correct sentence when decoding the cipher, then we can get more meaningful results.

7. Conclusion

In this project, we designed and implemented a genetic algorithm for solving the Z340 cipher. We assumed the cipher to be homophonic. We implemented two variants of crossover: simple crossover and intelligent crossover. Our technique used a dictionary lookup that helped to recognize English words from the decoded text. This gave more meaningful results than just looking at substring frequencies. We tested our algorithm on the Z408 cipher which has a known decrypt, and found that we were able to match 25% of the characters, without the use of any hints.

We ran a number of experiments on the Z340 cipher and observed the different words that were detected in the decrypts. We found many strong words such as "hate", "slain", "lethal", "kill", "stab" and "die" in many runs of the genetic algorithm. Particularly, the word "hate" appeared in most of the runs.

We observed that when we only consider substring scores, without using dictionary lookup, the words in the decrypt were mostly 3 or 4 letter words that were repeated many times. Furthermore, the decrypts seemed to be dominated by very few letters such as 't', 's', 'e' and 'a', which is not realistic in real English text. By considering dictionary lookup, we were able to obtain a wider range of words and a more realistic distribution of letters in the decrypt.

Appendix A: References

[1] Voigt, T. (March 20, 1998) *Zodiac letters and ciphers*. Retrieved December 22, 2007, from http://www.Zodiackiller.com/Letters.html.

[2] Donald L.Kreher , Douglas Robert Stinson(1999) *Combinatorial algorithms: Generation, Enumeration and Search.* CRC press New York.

[3] Wikipedia the Free Encyclopedia (March 2007) *Zodiac Killer.* Retrieved January 23, 2009 from http://en.wikipedia.org/wiki/Zodiac_Killer

[4] Kirps, J. (1996). *Cracking the Zodiac killer's 340 Cipher-Why Brute Force attack just won't work with the 340*. Retrived January 23, 2008, from http://www.kirps.com/web/main/ resources /various/Zodiac340.

[5] Stamp, M. (2005).*Information Security: Principles and Practice*. A John Wiley & Sons, Inc., Publications.

[6] Farmer, C. (June 10, 2007) *Zodiac*. Retrieved January 25, 2009 from http://www.opordana lytical .com/articles/Zodiac.htm

[7] Farmer, C. (March 06, 2007) *The Zodiac "My Name is Cipher".* Retrieved April 15, 2009 from http://www.opordanalytical.com/articles/Zodiac3.htm

[8] Farmer, C. (May 22, 2007) *The Zodiac 340 Cipher Solved.* Retrieved January 25, 2009 from http://www.opordanalytical.com/articles1/Zodiac-340.htm

[9] Dorigo. M, Maniezzo. V, and Colorni, A.(February, 1996) *The Ant System: Optimization by a Colony of Cooperating Agents*. Proceedings of IEEE Transactions on Systems, Man, and Cybernettics, 29-41. Retrieved March 6, 2008 from http://ieeexplore.ieee.org

[10] Flint,W. (February 16, 2001) *Plant your data in a ternary search tree*. Retrieved June 17, 2009 from http://www.javaworld.com/javaworld/jw-02-2001/jw-0216-ternary.html

[11] Singh, S. *Homophonic Cipher.* Retrieved November 5, 2008 from http://www.simonsingh .net/the_Black_Chamber/homophoniccipher.htm

[12] Dao, T. (December 2007) *Analysis of the Zodiac 340-Cipher*. Retrieved January 15, 2008 from http://www.cs.sjsu.edu/faculty/stamp/students/ThangDao_299_Report.pdf

[13] Montaldo, C. *The Zodiac Killer.* Retrieved February 23, 2008 from http://crime.about.com /od/history/a/Zodiackiller.htm

[14] Cole, M. (August 10, 2003). *Two new theories regarding the Zodiac Case*. Retrieved January 23, 2008 from http://www.mikecole.org/Zodiac/two_theories/1.2/

[15] Wikipedia the Free Encyclopedia. *Letter frequency.* Retrieved February 7, 2008 from http://en.wikipedia.org/wiki/Letter_frequency

[16] Tru_{TV} Crime library. *Criminal profiling*. Retrieved May 15, 2008 from http://www.trutv.com/library/crime/criminal_mind/profiling/index.html

[17] Zamora, H.J., San Francisco Chronicle Staff Writer (January 27, 2007). *1967-71 – a bloody period for SF police.* Retrieved October 27, 2009 from http://www.sfgate.com/cgi-bin/article.cgi?f=/c/a/2007/01/27/MNG9DNQ8TQ1.DTL

[18] Wikipedia the Free Encyclopedia. *Ternary search tree.* Retrieved June 15, 2009 from http://en.wikipedia.org/wiki/Ternary_search_tree

[19] Bentley, L.J., & Sedgewick, R. (1997). *Fast algorithms for sorting and searching strings.* Proceedingd of the 8th ACM-SIAM symposium on Discrete algorithms, 360-369. Retrieved June 20, 2009 from ACM Digital Library

[20] Goldberg, E.D. (1989). *Genetic algorithm in Search, Optimization, and Machine learning*. Addison –Weslwy Publications.

[21] Shannon, C.E. (October 1928). *A mathematical theory of communication*.Bell System Technical Journal, Vol. 27, 379–423, 623–656. Retrieved November 9, 2009 from http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf

[22] Wikipedia the Free Encyclopedia. *Cryptography.* Retrieved January 22, 2008 from http://en.wikipedia.org/wiki/Cryptography

[23] Oranchak, D. (July 2008). *Evolutionary algorithm for decryption of monoalphabetic homophonic substitution ciphers encoded as constraint satisfaction problems.* Proceedings of the 10th annual conference on Genetic and evolutionary computation, 1717-1718. Retrieved November 14, 2009 from ACM Digital Library.

[24] Olson, E. (October 2007). *Robust Dictionary Attack of Short Simple Substitution Ciphers.* Taylor & Francis, Inc. USA, Vol. 31, 332-342. Retrieved November 14, 2009 from ACM Digital Library.

[25] Sait, M.S., & Youssef, H.(1994). *VISI Physical Design Automation: Theory and Practice.* McGraw-Hill, Inc. USA. Retrieved November 14, 2009 from ACM Digital Library.

Appendix B: Interesting decrypts found for Z340 cipher

a) Z340 decrypt obtained without using any cribs

tejiyHITlcsdbcunkiaTOOnSENSEDhguitzSOFAwpcnoiDOTiahiihhyahi
oinaaeguvtFATESncmrTANwleiaehrynshnelrjueooMAGeseuWONcietdco
cuauSONavkSTABdtndFADAPEurtozHEWgSTYwiAHAanihdrtzswaeeMETcES
TidbfGNUdoYEAchmbmfeANTrtethzOAKcmrTOOeaeycrkHUMumdeoimeaazt
nshswtetjiwpmSACoeottgoniheufeCUEoBADmuedaryntcidaaiNETNOTnh
ibeoiicrlnndhetvsnthseojfeikilhtpnahcgea

Strong words like HIT, SENSED, FATES, SON, STAB, BAD can be seen in this decrypt.

b) Z340 decrypt obtained using the crib (HER, ZODIAC)

The words found in the solution are:

her	here	luna	ate
put	silts	sit	old
pan	ids	ton	mid
hour	crim	crimp	ice
slit	salt	sir	life
cop	cops	ail	net
tit	leat	fail	failed
son	reel	reels	sore
all	hems	sad	tin

zodiac
printing decrypt with words found in capital letters:
HEREiltLUNAATEePUTiilstoiilnasocSILTSioiszelSITtihkeiiatldOL
DioiircPANoiIDSuaeauiTONMIDrtaloHOURnaepisteiCRIMPsICElaitas
epksstoiatniiktnusoifizrpalitilochilSLITkioiaSALThSIRmedeain
LIFEoCOPSilrkaoeeeoaitlAILNETTITaealismdrLEATtsepeFAILEDiitl
caoSONREELSzehiaiadilcsoitaSOREprieiteprsiALLlaitiiilriliuoe
HEMSiiaanulSADlanolohrieoihTINeiZODIACmi

Words like HOUR, SIR, LIFE, COPS, FAILED, SAD can be seen in this decrypt. We can also see a word CRIMP which is close to the word CRIME.

c) Z340 decrypt obtained using the words in the known Z408 decrypt as the dictionary

(i) Only simple crossover –

Number of iterations: 22
The words found in the solution are:

man the her
the than hate
Printing decrypt with words found in capital letters:
reteetritatlnishnieeenehhaelieaenheseanearsenhneirieeelrthai
neheetehetneieintthteanalehtrhthMANsahehaneteeshshaansiiintn
shinieheeilieantneneterthheasannemitaieriehelehesmaetlteetal
eetinehheattitatitnieaehinteseeitTHEanshttshirnthttiaiteeese
nlaentseeiartmetainieenHERinnsshtaienthseehteethneehesieaTHE
rilnheTHANeeleeelheamsaenarieaeirhHATEse

(ii) Only intelligent crossover:

Number of iterations: 56
The words found in the solution are:
her not the her
her that the not
hate all
printing decrypt with words found in capital letters:
iianetnrnahiotreineeaonhaanainttftainiheHERNOTonlicleeinthtr
teheeateruhelthiheenelaanehaneTHEtitaeneaoneetaahehierriloho
recfhnHERnalerouinHEReeaeeaiiTHATelthrenceheineaieheanetihaa
aerthTHEnitachtetehielaelhulinenheeaiohhatrennfeeeriireteeia
eitnautinrheeeehiitlatohenifhareaiteoeetneetnahtoeetntlninhl
itNOTeheainnitaraHATEtinhaineALLehhththe

(iii) Simple and intelligent crossover:

Number of iterations: 50
The words found in the solution are:
the her her the
hate all not all
printing decrypt with words found in capital letters:
eataTHEthointainnttennseelehnoaaiinesolearienitseetettteheat
tteeetancileeheniliHERhooteteiHERanloianlnslealesnaoiitnetin
intieseecTHEentinoleaertninoevthareHATEeteettoineraetolhailh
ntaalaenoohttialALLnernietieesetilinonsethiiteilnlaNOTlheeen
inashilaatarlreiontenanetenillintoaetlnloeiheniiteeieleeohee
eaonitiioneothnchenarloALLettoeereeviase

We notice words like THE, HER and HATE to appear during most of the runs.

Appendix C: Interesting decrypts found for Z408 cipher

a) Z408 decrypt obtained using the crib (I, KILLING)

(i) iLIKEKILLINGezcelanonpithincoimsinimihieiosrjstiaeiomKILLING
rildgysaihwiemvdqztnnanoioospaiieihscneoomgsjeiaynisplrtollw
vKILLocsaniihggileiszeiasmnewidilliagoxehqsmnainitelznnaeeoj
wiyhghneiagtridqvnkocmtrieingijlwisnainepdemmieinwiysriemioi
zirillnaqonrjniheodydinhoaoplleisiiolaKILLEDrillnzncsasttlnl
ooirillnregilotvisthosanenpiiatririllwjtnmtlciocravqowmestnv
llhneingcttlpleomrjstnmeodlitahnacdiaehsaniieiei

(ii) iloktKILLINGltpllohehersuiipstserhszreiuiyedcoerlrqpnKILLong
WILLgseziebstztmhtsihohyrseeelituquePARTpngocurzsnieeldepllb
tkollspezisieggivttetuqoesarbsmillolgexluhonhziiistvtnhourec
bqseguiuilgndrmhthkspzeworsagiclbqohztilemusziriabssowqtNOTi
tiWILLhohehdcnoelpmslihuyltellusoiqpvzKILLtlwollhthpeoenSLAV
ESiWILLndrGIVEntrenepeohTHErtzndrWILLbcnisslpitpwlthybslenht
Lluhrongpeslevtszdcenazuemloeouhzpmiouuezisqliui

Words like LIKE, GIVE, WILL and SLAVE are observed many times in the Z408 decrypt when the crib {I, KILLING} is used.

b) Z408 decrypt obtained using the crib (I, KILLING, SLAVES, GIVE)

The words found in the solution are:

like	kill	killing	kill
killing	all	kill	give
the	even	girl	all
have	kill	slaves	give
all	slaves	the	

printing decrypt with words found in capital letters:

iLIKEKILLINGeeheleeeeaasnithstinaeteanieicnerteaothahKILLINGrilnganeineteealeesteeecasen
aoitehnnhattahgtreaeaninaleeALLeaKILLshnettingGIVEtneeheniatetlilliogeleneTHEeitisEVENe
eeterehangnteiogeealeaeksheerittaGIRLehteettealeieitiaetatrhehitieirilleeeeeerninealaniencotALL
ettiHAVEKILLenrilleeehneneSLAVESirillnetGIVEeaanenaneeeeaateeearilleretislhithroaeceiene
eALLnetingheSLAVESeerneaeeellieeneehlieennetTHEiei

Solution has 201 characters matching with the Z408 decrypt

Appendix D: Zodiac cover letters

a) Zodiac 408 cover letter

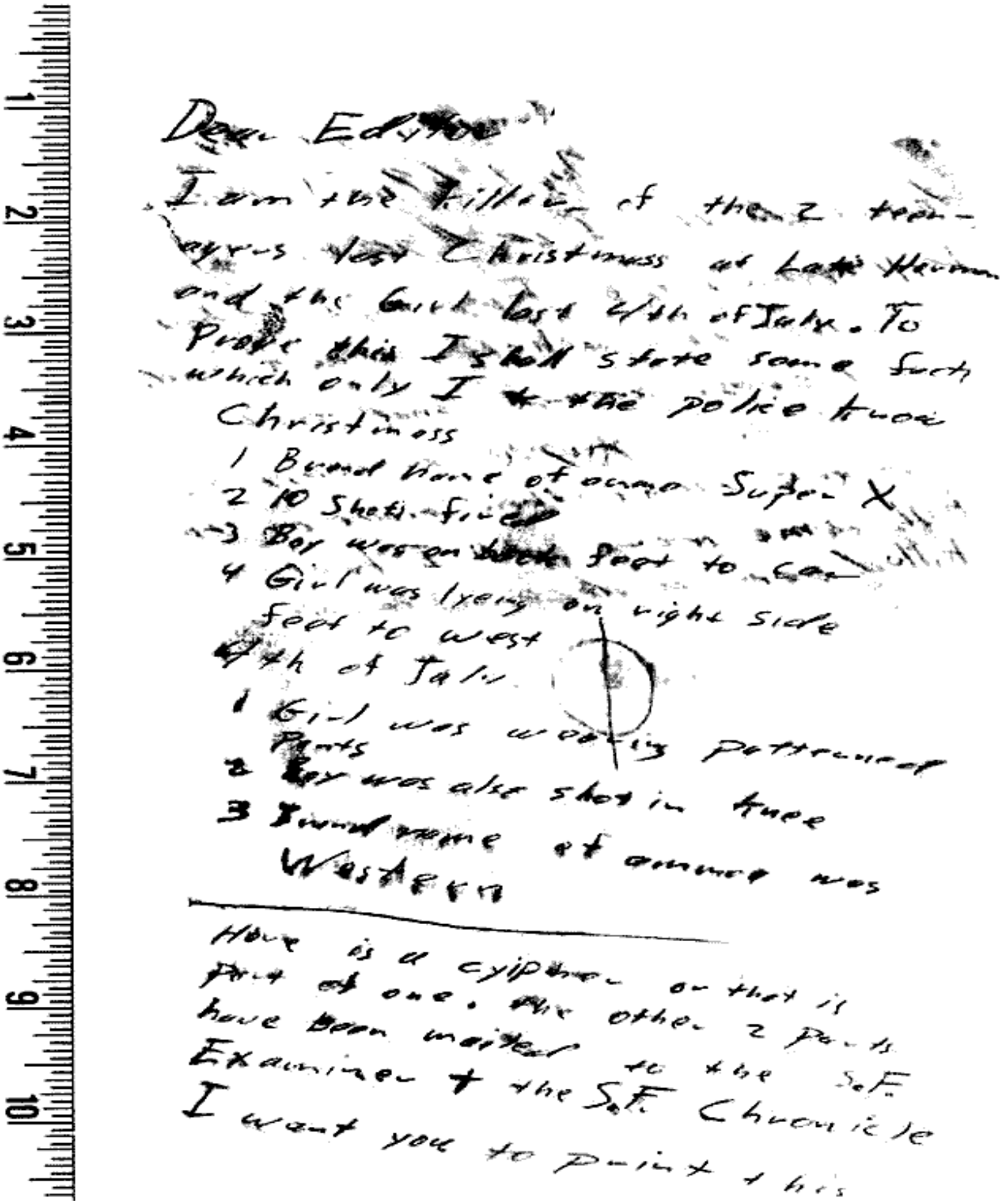
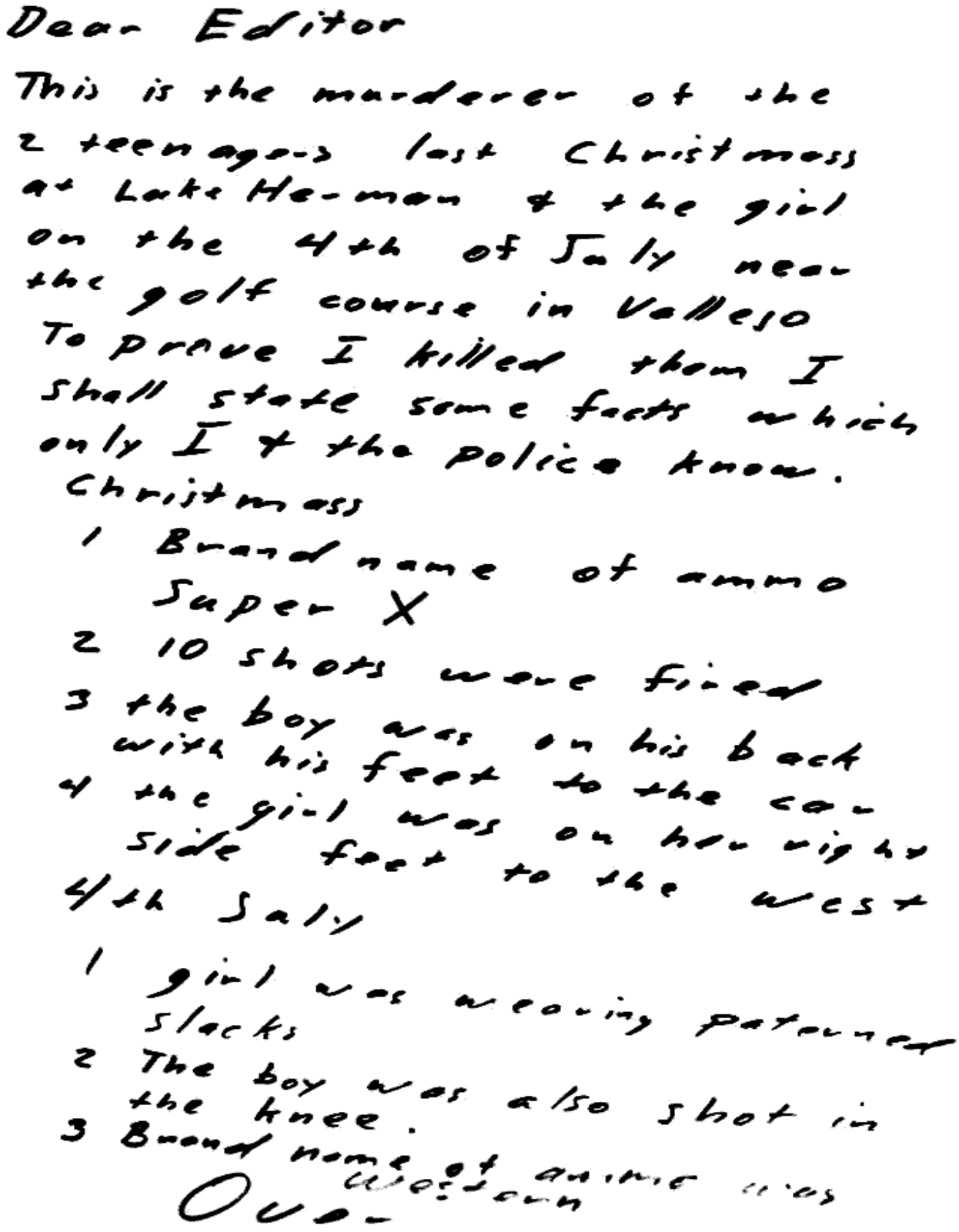


Figure 22: Z408 part 1, July 31, 1969 Times Herald letter


Dear Editor

This is the murderer of the 2 teenagers last Christmass at Lake He-man & the girl on the 4th of July near the golf course in Vallejo To prove I killed them I shall state some facts which only I & the police know.

Christmass

1 Brand name of ammo Super X

2 10 shots were fired

3 the boy was on his back with his feet to the car

4 the girl was on her right side feet to the west

4th July

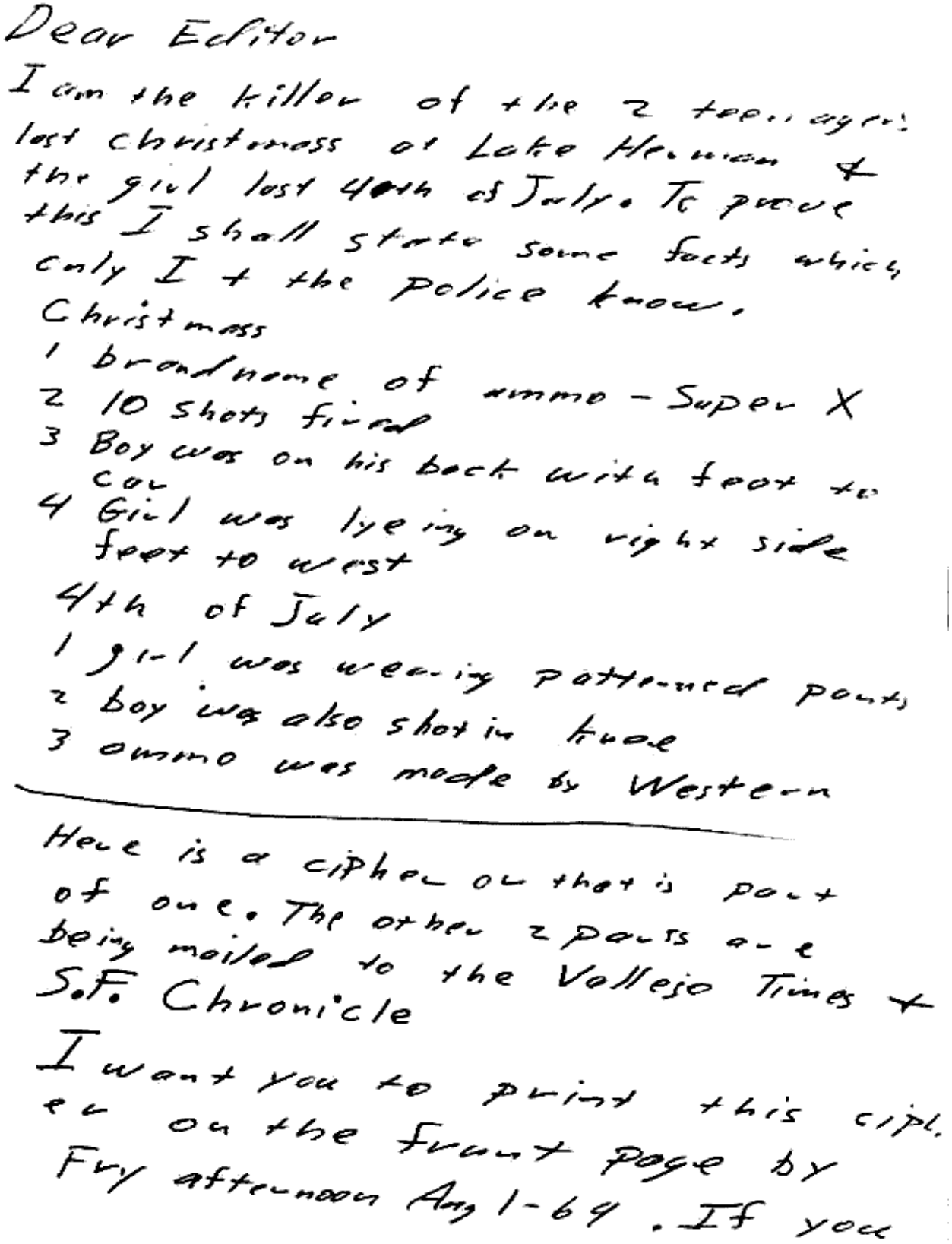
1 girl was wearing patterned slacks

2 The boy was also shot in the knee.

3 Brand name of ammo was Western

Over-

Figure 23: Z408 part 2, July 31, 1969 Chronicle letter



Dear Editor
I am the killer of the 2 teenagers last Christmass at Lake Herman & the girl last 4th of July. To prove this I shall state some facts which only I & the police know.
Christmass
1 brand name of ammo - Super X
2 10 shots fired
3 Boy was on his back with feet to car
4 Girl was lying on right side feet to west
4th of July
1 girl was wearing patterned pants
2 boy was also shot in face
3 ammo was made by Western

Here is a cipher or that is part of one. The other 2 parts are being mailed to the Vallejo Times & S.F. Chronicle
I want you to print this cipher on the front page by Fry afternoon Aug 1-69. If you

Figure 24: Z408 part 3, July 31, 1969 Examiner letter

b) Zodiac 340 cover letter

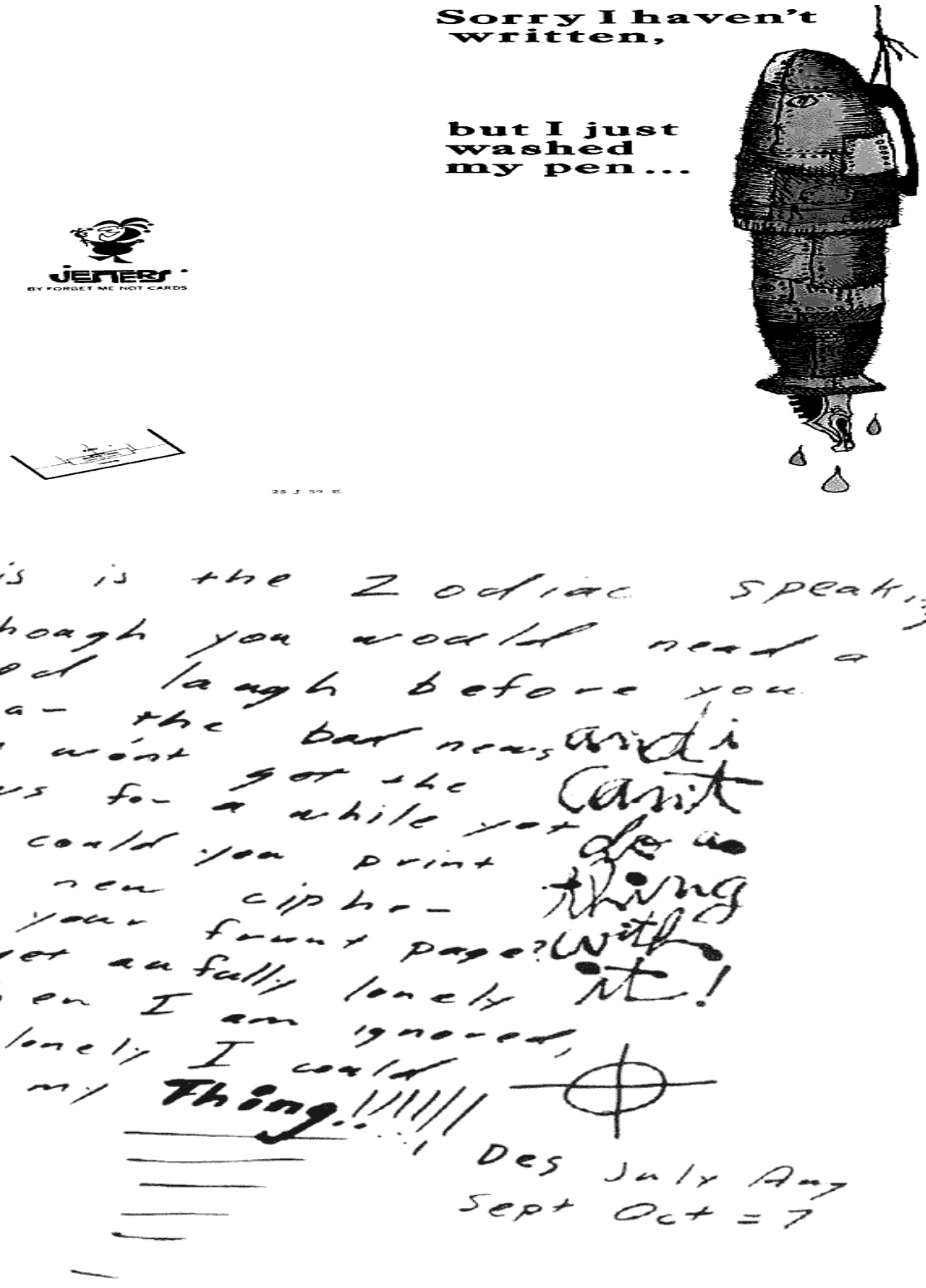


Figure 25: Z340 cover letter

c) Z13 / My Name Is …. Cover letter and Bomb Diagram

This is the Zodiac speaking
By the way have you cracked
the last cipher I sent you ?
My name is ——

A E N ⊕ ⊗ K ⊗ M ⊗ ⊥ N A M

I am mildly cerous as to how
much money you have on my
head now. I hope you do not
think that I was the one
who wiped out that blue
meannie with a bomb at the
cop station. Even though I talked
about killing school children with
one. It just wouldnt do to
move in on someone elses territory.
But there is more glory in killing
a cop than a cid because a cop
can shoot back. I have killed
ten people to date. It would
have been a lot more except
that my bas bomb was a dud.
I was swamped out by the
rain we had a while back.

Figure 26: Z13 cover letter

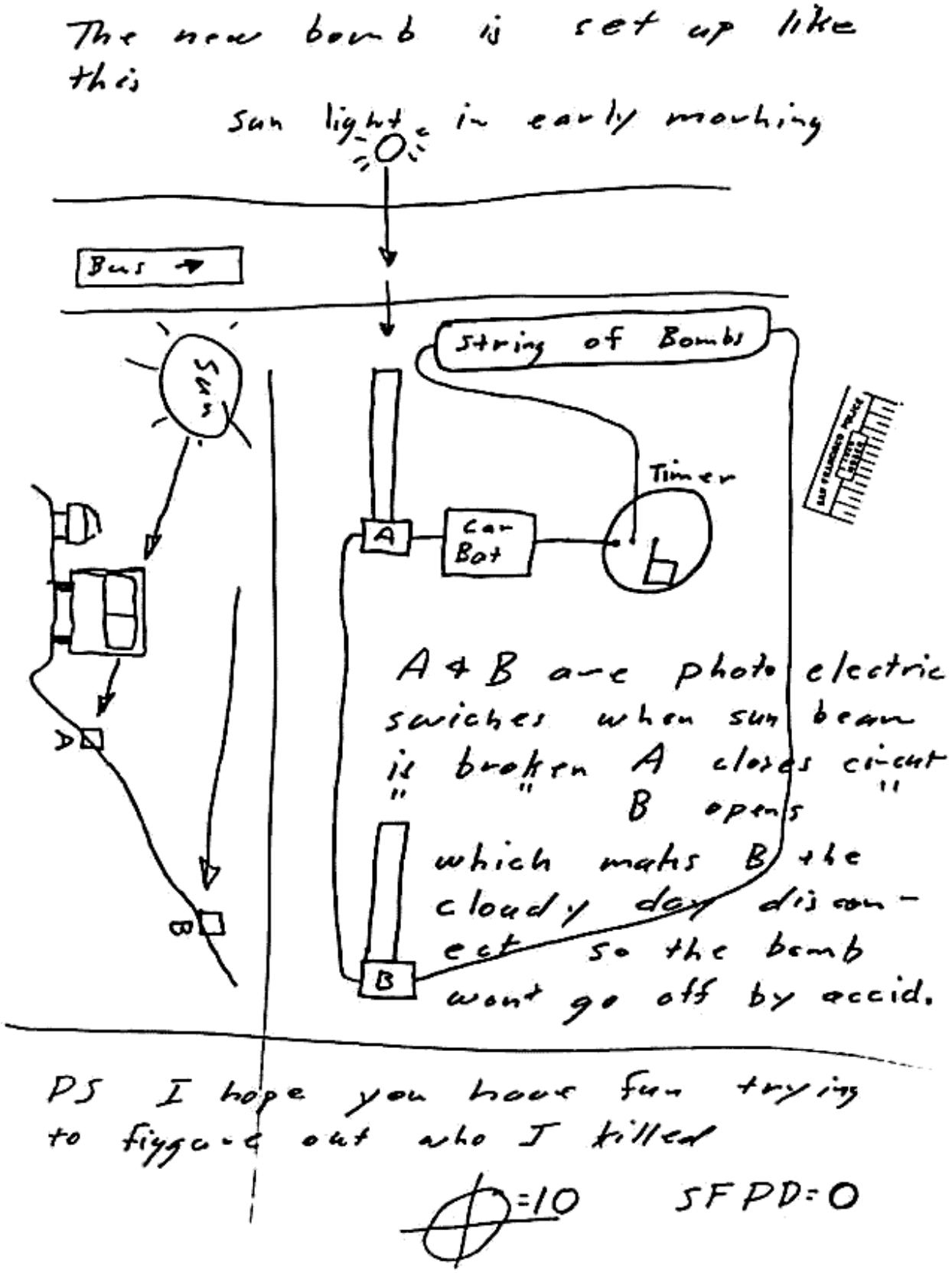


Figure 27: Z13 Bomb Diagram

d) Z32 cover letter

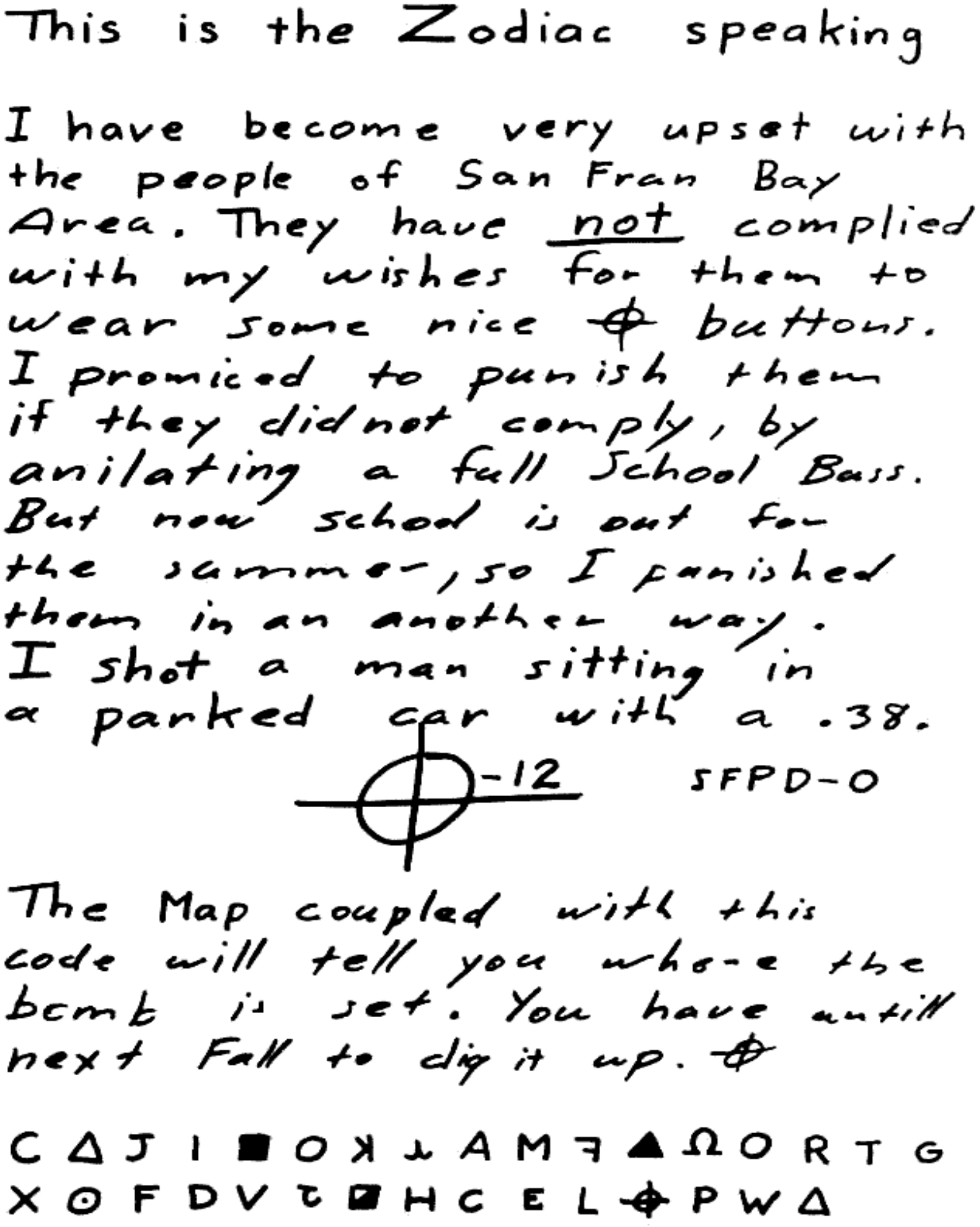


Figure 28: Z32 button cover letter